Trabajo Fin de Máster
"Máster Universitario en Microelectrónica: Diseño y Aplicaciones de Sistemas Micro/Nanométricos"

Diseño de algoritmos de procesado de imágenes médicas en tiempo real para su implementación en la plataforma Raspberry Pi

Autor: Rubén Padial Allué

Tutores: Juan A. Leñero Bardallo y José Bernabeu Wittel

7 de septiembre de 2020

Índice

Índice de figuras

Índice de tablas

Resumen

El objetivo de este Trabajo Fin de Máster es la implementación en código Python de una interfaz de usuario para visualizar y procesar imágenes médicas en la banda infrarroja para el estudio clínico de anomalías vasculares. La interfaz tiene varias opciones de procesamiento de imagen en tiempo real para obtener información relevante desde un punto de vista médico. Se representan en tiempo real la termografía acompañada del gradiente de la imagen y otra imagen con líneas isotermas que permite delimitar el área afectada por la anomalía vascular. Además, ha sido diseñada teniendo en cuenta las prestaciones de un sistema basado en la plataforma Raspberry Pi.

Es trabajo forma parte de un proyecto de la Universidad de Sevilla en colaboración con el Departamento de Dermatología Pediátrica del Hospital Virgen del Rocío de Sevilla. El objeto de este proyecto es la implementación de un dispositivo portátil para el diagnóstico y seguimiento de malformaciones vasculares. Las malformaciones vasculares malignas se caracterizan por un incremento de la temperatura en la zona afectada consecuencia del alto flujo sanguíneo en dicha zona. Los sensores infrarrojos son capaces de medir este incremento de temperatura a través de la radiación. Las opciones presentes en el mercado limitan la implementación de algoritmos que ayuden en el diagnóstico. En este proyecto se plantea el uso de una Raspberry Pi para este propósito. Se trata de un ordenador de bajo coste muy versátil, al que se le pueden añadir diversos dispositivos y en el que se pueden implementar y adaptar los propios algoritmos.

La interfaz software desarrollada que incluye los algoritmos para el diagnóstico se va a incorporar al sistema existente y ser evaluada de forma clínica. Con las funcionalidades desarrolladas se espera tener una mayor capacidad de análisis de los datos en menor tiempo.

1 Introducción

En este capítulo se van a describir los tipos de anomalías vasculares más comunes, sus características y los métodos empleados para su diagnóstico. Algunas de estas anomalías son objeto de estudio del proyecto que de investigación en el que se encuadra el trabajo fin de grado.

1.1 Antecedentes

Las anomalías vasculares son anomalías difíciles de diagnosticar y evaluar debido a su similitud entre los distintos tipos. En el espectro visible todas las anomalías que se van a describir a continuación son muy parecidas a simple vista. Esto hace necesario derivar el paciente a pruebas especiales. Actualmente los métodos más utilizados para el diagnóstico y seguimiento de estas enfermedades son las ecografías de efecto Doppler y las imágenes de resonancia magnética (MRI) [7].

Las **imágenes por resonancia magnética** o **MRI** se consiguen aplicando un campo magnético que alinean los polos de las partículas del cuerpo. Las moléculas alineadas, típicamente hidrógeno, son excitadas por pulsos de radio-frecuencia (RF) que hacen que entren en resonancia y giren sobre sí mismas. La frecuencia se conoce como frecuencia de *Larmor*. La cantidad de energía que absorben las partículas, depende del tipo de energía y de frecuencia. Así se puede excitar las partículas de interés. Cuando el pulso termina, las partículas recuperan la posición de equilibrio emitiendo que induce una corriente. El tiempo que las partículas tardan en volver al estado de mínima energía

depende del tipo de partícula. Con estos tiempos se diferencian los tipos de partículas, asociándolos a intensidades en la imagen [23].

Para obtener una imagen con la ayuda del efecto **Doppler** se proyectan ondas de ultrasonidos (entre 1 y 10 MHz) en el cuerpo. Estas ondas rebotan en el tejido y son recibidas por un piezoeléctrico que reacciona al estímulo generando una diferencia de tensión en sus terminales. El tiempo que pasa entre la onda emitida y el eco recibido da una medida de la profundidad del tejido. Los datos obtenidos se procesan y se genera la imagen. Este método es usado normalmente porque no somete al paciente a ningún tipo de radiación [23].

En el caso de la MRI, el equipo es muy caro y poco accesible, lo que resulta en la mayoría de los casos en tiempos de espera muy largos. Los resultados de esta prueba tampoco son inmediatos. Para las ecografías los equipos son más asequibles, pero, al igual que la resonancia magnética, requiere un equipo especializado para adquirir la imagen e interpretarla. Ambas técnicas son complicadas para aplicar en niños. Hacer una resonancia magnética implica introducir al paciente en un aparato con reducido espacio con un alto ruido. Para adquirir la imagen de la ecografía es necesario el uso de geles y el contacto físico. En ambas técnicas los resultados no se obtienen de forma inmediata, si no que se necesita un tiempo para analizar los datos y obtener un resultado concluyente. Estos aspectos hacen que sea difícil evaluar las malformaciones en niños de forma rápida. Esta la motivación inicial del proyecto: crear un dispositivo de bajo coste que permitiera realizar un diagnóstico claro y temprano en la clínica.

La clasificación de las distintas anomalías vasculares no es objeto de este trabajo. La distintos tipos de malformaciones vasculares ha sido agrupadas por diversos autores en 2 grupos: de bajo flujo y de alto flujo [4] [5] [6]. Las malformaciones de bajo flujo son malformaciones den venas, capilares o en los canales linfáticos. Las malformaciones de alto flujo afectan a las arterias y son las más graves y delicadas. Se denominan de alto flujo por el alto flujo sanguíneo que hay en ellas. Diversos estudios [1] [2] [3] ha demostrado que hay un incremento de temperatura en las malformaciones vasculares de alto flujo respecto a las zonas cercanas de la piel.

Desde los inicios de la medicina en la antigua Grecia, Hipócrates pudo relacionar que los tejidos afectados por alguna enfermedad con un incremento de la temperatura de los mismos [1].

Hasta mediados del siglo XX los médicos se valieron de este conocimiento para diagnosticar enfermedades con sus manos, palpando las diferentes zonas del cuerpo del paciente en busca de diferencias de temperatura. Es en 1956 cuando Lawson introduce el uso de la termografía infrarroja en la medicina. Esta tecnología se había utilizado en la industria pero no se había usad en la medicina [1].

El cuerpo humano transfiere calor por conducción convección, evaporación y radiación. En condiciones estables de temperatura entre 18 y 25 ºC el principal mecanismo para transferir calor es la emisión de radiación infrarroja en un rango entre 8-10 µm. Las termografías infrarrojas son capaces de adquirir radiación infrarroja entre los 0.8 µm y 1 mm, pudiendo transformar estos niveles de radiación adquiridos en temperatura [1].

Las diferencias en el flujo sanguíneo dan lugar a un cambio de temperatura que a su vez resulta en una radiación infrarroja transmitida. En varios estudios se concluye que la termografía es un método fiable y muy útil para el diagnóstico y seguimiento de estar enfermedades al tratarse de un método no invasivo, seguro, indoloro y rápido [1] [2] [3]. Al tratarse de un método no invasivo, se puede repetir sin miedo a posibles efectos secundarios [1]. Estas características hacen que la termografía infrarroja se presente como un excelente método de diagnóstico para pacientes infantiles [3].

En el estudio "Thermography for the differential diagnosis of vascular malformations" [27] el profesor J. A. Leñero, tutor de este Trabajo Fin de Master, entre otros compañeros utilizan imágenes termográficas para diferenciar malformaciones vasculares de alto y bajo flujo. En la Figura 1.1 se comparan 2 anomalías aparentemente iguales en la mano de un paciente. En la Figura 1.2 se hace lo propio con anomalías en la espalda que a simple vista pueden parecer del mismo tipo. En ambos casos se llega a la misma conclusión: Las malformaciones vasculares de alto flujo generan un incremento de la temperatura en la zona afectada, probando la fiabilidad de esta técnica para el diagnóstico.

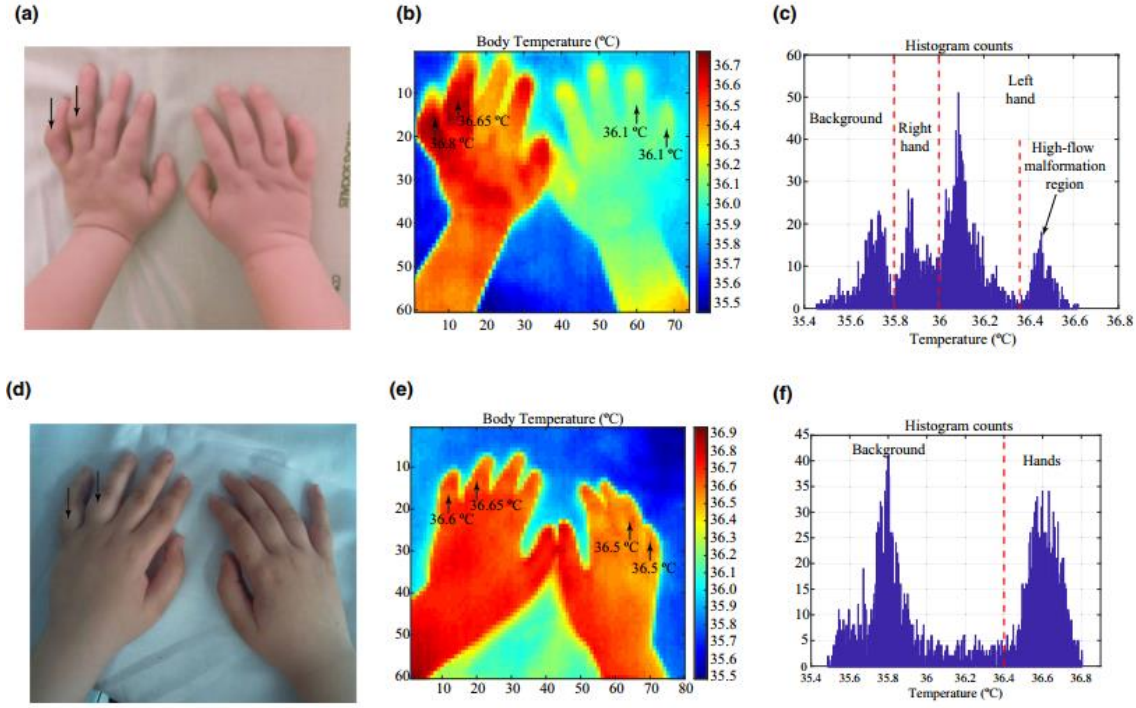


Figura 1.1. (a- f) Comparación entre malformaciones de alto flujo (fila superior) y bajo flujo (fila inferior) en la mano. (a y d) imágenes en el espectro visible. (b y e) imágenes termográficas. (c y f) histogramas de las imágenes termográficas.

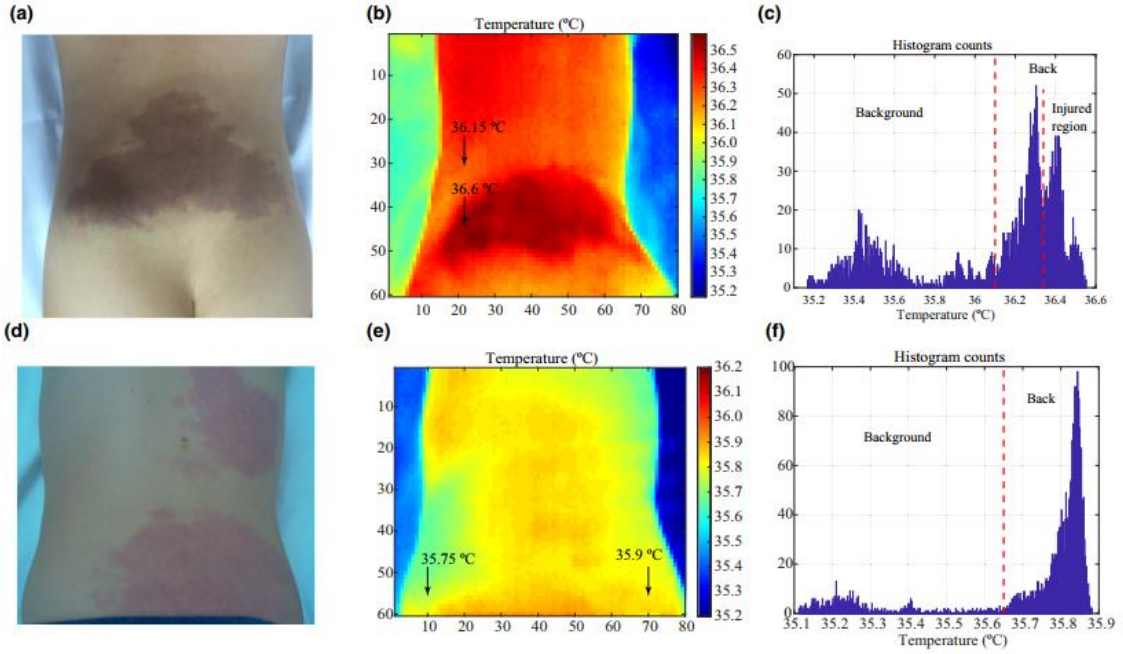


Figura 1.2. (a- f) Comparación entre malformaciones de alto flujo (fila superior) y bajo flujo (fila infrerior) en la espalda. (a y d) imágenes en el espectro visible. (b y e) imágenes termográficas. (c y f)histogramas de las imágenes termográficas.

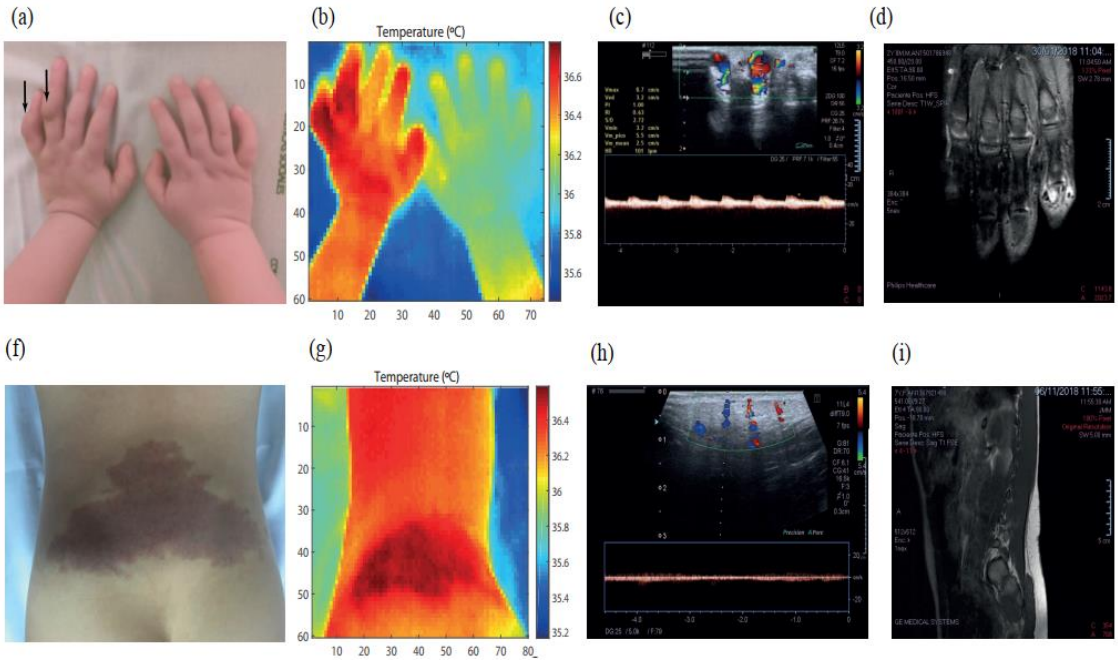


Figura 1.3. Comparación entre las diferentes técnicas de imagen médica en malformaciones vasculares de alto flujo. De izquierda a derecho: imagen en el espectro visible, imágenes termográficas, imágenes obtenidas con ultrasonografía Doppler e imágenes de resonancia magnética.

En otro estudio del profesor J. A. Leñero [10] se compara la técnica de las termografías con las ecografías de efecto Doppler y la resonancia magnética. En la Figura 1.3 se muestran las imágenes del estudio. Comparando las termografías y las ecografías de dos malformaciones de alto flujo, el estudio concluye en que ambos métodos proporcionan resultados similares: ambos son capaces de identificar las zonas en las que el flujo sanguíneo es mayor, bien gracias al efecto Doppler o a la radiación generada por

el incremento de temperatura en la zona afectada. Las imágenes obtenidas por resonancia magnética ayudan en la evaluación de la extensión y profundidad de las malformaciones, siendo un método complementario. El estudio también llega a la conclusión que las técnicas tradicionales necesitan equipos avanzados, personal especializado y un mayor tiempo para obtener los resultados. Esto juega favor de los métodos que utilizan imágenes infrarrojas, más rápidos y fácil de interpretar.

1.2 Planteamiento del problema

La evolución en la tecnología hace que cada vez sea más sencillo y económico desarrollar dispositivos portátiles más complejos. Esto nos permite tener a nuestro alcance dispositivos con funciones que antes no eran accesibles.

Raspberry Pi se ha convertido en una plataforma extensamente usada para dispositivos portátiles y en IoT. Se trata de un ordenador de tamaño y precio reducido. Si le sumamos el descenso del tamaño y precio en los sensores de imagen infrarroja, en sencillo pensar en un dispositivo portátil “low-cost” que adquiera y procese imágenes infrarrojas.

En el mercado se encuentran distintas posibilidades de dispositivos de imagen. Muchos ellos tienen una alta resolución de imagen y combinan sensores de imagen infrarrojos y RGB. Estos sistemas están diseñados para un uso genérico y el usuario no puede modificar los algoritmos y adaptar el sistema para otro uso distinto [10]. También hay disponibles cámaras IR que se pueden conectar a los smartphones actuales. Estos dispositivos cuentan con aplicaciones propias y están muy limitados en cuanto a su uso ya que no se pueden añadir periféricos ni programar los propios algoritmos.

Existen otros dispositivos con sensores de imagen infrarrojos específicos para aplicaciones médicas, pero no cuentan con sensores para adquirir imágenes en el espectro visible al mismo tiempo. Además, necesitan estar conectados a un ordenador a través de USB. Esto hace que no sea un sistema fácil de ser transportado [10].

Aunque la resolución de la imagen es menor que en los dispositivos comerciales, la elección de la Raspberry Pi permite la flexibilidad de añadir periféricos según las necesidades con un bajo coste. Se puede adaptar a diferentes aplicaciones con la elección de unos dispositivos u otros y cambiado los algoritmos del procesamiento. Así por ejemplo este dispositivo que inicialmente está ideado para la detección y evaluación de malformaciones vasculares podría adaptarse por ejemplo en la actualidad para la detectar casos de la COVID-19 e incluso realizar un seguimiento [10]. Al tratarse de un dispositivo de bajo coste y con una plataforma fácil de customizar facilitaría la extensión de su uso, siendo accesible para casi cualquier centro médico.

1.3 Anomalías vasculares

Las anomalías vasculares, también llamadas *angiomas* o *hemangiomas*, comprenden a los tumores vasculares y a las malformaciones vasculares. Estas pueden ser desde una marca de nacimiento hasta trastornos que pueden ser un riesgo para la salud, que afectan principalmente a bebes, niños y jóvenes. Dada esta gran variedad es muy importante diagnosticarlas, clasificarlas y tratarlas correctamente [5].

En 1863, Virchow hizo la primera clasificación únicamente diferenciando los angiomas en simples, cavernosos o ramosos. los patólogos encontraron una variedad cada vez más amplia de patologías vasculares y, al no poder clasificar las lesiones dentro de los sistemas existentes, crearon nuevas clasificaciones y nomenclatura. En 1982, Mulliken y Glowacki propusieron una clasificación de las anomalías vasculares de acuerdo con las características endoteliales. En este sistema binario, los hemangiomas se caracterizan por una mayor proliferación e hiperplasia, mientras que las malformaciones demuestran displasia de vasos con tasas normales de renovación celular [6].

Uno de los objetivos principales de ISSVA ("International Society for the Study of Vascular Anomalies") es la clasificación uniforme de las anomalías vasculares. Se han hecho varias revisiones de la clasificación de estas enfermedades. En la clasificación de 1996 se separaron las anomalías vasculares en 2 grupos de acuerdo a las clasificaciones anteriores: tumores vasculares y malformaciones vasculares. Esta clasificación se hace por el comportamiento de las enfermedades: los tumores vasculares tienen a propagarse, mientras que las malformaciones vasculares suelen ser estáticos [5].

En 2014 se hizo una actualización para hacer la clasificación más flexible y para añadir información genética, patológica e histológica. En 2018 se hizo una revisión de la clasificación de 2014. En la Figura 1.4 se muestra de forma esquemática la clasificación de 2014 de acuerdo a la revisión de 2018 [5] [6] [8].

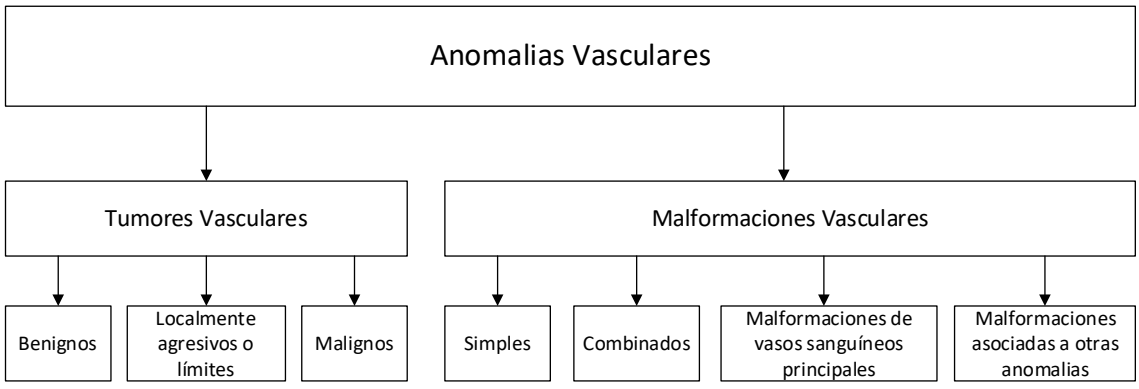


Figura 1.4. Clasificación de las anomalías vasculares de acuerdo a la revisión de 2018 de ISSVA.

1.3.1 Tumores vasculares

Los tumores vasculares se caracterizan por la hiperproliferación de células endoteliales. Esto quiere decir que los tumores vasculares son aquellas enfermedades vasculares en las que tejidos que revisten interiormente algunas cavidades orgánicas que no comunican con el exterior, como es el caso de los vasos sanguíneos, se multiplican. Normalmente no son congénitas, se desarrollan después del nacimiento, y suelen crecer rápidamente. Se pueden clasificar basándose a su comportamiento celular en benignos, localmente agresivos o límites, o malignos [6].

(a) *(b)*

Figura 1.5. (a) Hemangionma infantil en la zona cervicofacial. (b) hemangioma infantil en la cabeza.

Dentro de los tumores benignos, más común es el *hemangioma infantil*, con una incidencia estimada de entre un 4 y un 10 % entre todos los bebes y niños [5] o entre un 4 y un 5% según otras fuentes [6]. Es más frecuente en el género femenino y en las zonas cervicofaciales. Suelen aparecer en las primeras semanas después del nacimiento como una lesión cutánea o en los tejidos blandos que crece durante meses para después decrecer gradualmente sin necesidad de intervención [5] [6]. En la Figura 1.5 (a) se muestra un hemangioma infantil en la zona cervicofacial de un paciente del Hospital Virgen del Rocío de Sevilla. En la Figura 1.5 (b) se puede ver otro hemangioma infantil, esta vez en la cabeza, de un paciente del hospital sevillano. Ambos casos se estudiarán en la sección 4 junto con otros ejemplos. En la Figura 1.6 se ven dos imágenes (a) (b) correspondientes a hemangiomas infantiles en la zona superior de la espalda adquiridas en dos sesiones distintas.

(a) (b)

Figura 1.6. (a) y (b) hemangioma en la zona superior de la espalda en 2 sesiones.

Los *hemangiomas congénitos* son mucho menos comunes que los infantiles. A diferencia de la mayoría de tumores vasculares, estos están completamente formado en el nacimiento. Normalmente son lesiones cutáneas, pero pueden afectar también al hígado o a otros órganos. Las lesiones son de forma ovalada o redonda, pero clínicamente heterogéneas, y aparecen como placas de color rosa a violeta [6]. Suelen desaparecer rápidamente, antes del primer año de edad, también pueden permanecer de forma estable o desaparecer parcialmente [5].

Otro tipo son los *angiomas en penacho*, que aparecen como placas o máculas rojizas o marrones. Hay muy pocos casos de esta enfermedad presente en el nacimiento. Algunas lesiones desaparecen espontáneamente, especialmente en casos congénitos. están compuestos por pequeños mechones de capilares, característicamente rodeados por un vaso en forma de hendidura creciente en la dermis y regiones subcutáneas, con una distribución de bala de cañón [5].

El más común entre los tumores localmente agresivos es el *hemangioendotelioma kaposiforme.* El aspecto es muy similar al angioma en penacho con la diferencia de que suele tener un área más amplia [5]. A diferencia del tumor benigno de apariencia similar, puede afectar a tejidos musculares y adiposos [6], por lo que es considerado como localmente destructivo.

Los tumores vasculares malignos incluyen los *angiosarcoma* y *hemangioendotelioma epitelioide* entre otras enfermedades. Los angiosarcomas suelen surgir en los tejidos blandos subcutáneos de la cabeza, cuello y pecho, y comúnmente aparecen entre la 5ª y 7ª década de vida. La mayoría de los casos son esporádicos aunque pueden estar asociados con tratamientos de radiación previos o linfedemas crónicos [6].

1.3.2 Malformaciones vasculares

Las malformaciones vasculares se dividen en 4 grupos: simple, combinadas, de vasos sanguíneos principales y asociadas con otras anomalías. A diferencia del comportamiento de los tumores, que suelen crecer y desaparecer rápidamente, las

malformaciones normalmente crecen lentamente [6]. Se cree que las malformaciones vasculares son el resultado de errores de desarrollo del embrión en las células vasculares [7] de ahí que suelan estar presentes desde el nacimiento, aunque a menudo no se aprecian claramente desde el inicio y con el tiempo son más visibles. Pueden afectar a cualquier tejido del cuerpo, pero los tejidos más comunes son la piel, tejidos blandos, huesos, órganos o articulaciones [6].

- Malformaciones vasculares simples

Dentro de las malformaciones vasculares simple se encuentran las formadas por un único tipo de vaso sanguíneo (capilares, linfáticos, o venas) con las excepciones de las malformaciones arteriovenosas (AVMs) y las fistulas arteriovenosas (AVFs), que contienen arterias, venas y capilares [5].

Las *malformaciones de capilares* suelen afectar a la piel y a la mucosa, manifestándose como una mácula con tonos que van entre rosáceos y rojizos. Se encuentran desde el nacimiento y normalmente se suelen mantienen de por vida. [5] Pueden hacerse más grandes y oscurecerse con el tiempo. Normalmente se suelen confundir las malformaciones capilares con las comúnmente conocidas como "manchas de vino" (PWS) aunque hay otros tipos como la mancha de salmón (*nevus siplex*) coloquialmente conocidas como "mordisco e cigüeña o "beso de ángel". A diferencia de las manchas de las primeras, estas malformaciones desaparecen en los 2 primero años de vida. Estos 2 tipos son difícil diferenciables, aunque las manchas con un tono más rosáceo, con los bordes no definidos y en las regiones centrales suelen ser manchas de salmón [6]. Las PWS suelen afectar a las regiones faciales [5] [6] [7].

(a) *(b)* *(c)*

Figura 1.7. (a) Malformación capilar macular.(b) Nevus simlex. (c) Mancha de vino.

Para diagnosticar estas enfermedades se suele recurrir a la ayuda de imágenes de resonancia magnética [7]. En la Figura 1.7 [5] 67[6] se muestran ejemplos de malformaciones capilares en bebés.

Otro tipo son las *malformaciones venosas* (VMs). Cuando la lesión es superficial se muestran como una decoloración azul de la piel. Pueden afectar a tejidos blandos supútatenos y a cualquier tejido o víscera. Son suaves y se pueden comprimir. Tienden a aumentar su volumen con un aumento de la presión venosa. Estas malformaciones pueden provocar trombosis debido al lento flujo de la sangre en

ellas. Las malformaciones comunes suelen estar presentes desde el nacimiento, aunque las enfermedades intramusculares se dan después provocadas por un ejercicio físico muy exigente. Las malformaciones venosas genéticas se manifiestan como múltiples lesiones en piel y mucosa [5]. En la Figura 1.8 [5] se muestra una malformación venosa alrededor del labio y cuello con su respectiva imagen obtenida a través de resonancia magnética (A y B) y otro ejemplo de malformación venosa.

Las inicialmente estas malformaciones se detectaban con ultrasonidos y ecografías Doppler, basándose en identificar flujos de sangre bajos. Las imágenes de resonancia magnética son más útiles para diagnosticar el grado de estas lesiones. Cuando las malformaciones venosas son pequeñas y profundas, las angiografías son especialmente útiles para definir el grado de la lesión [7].

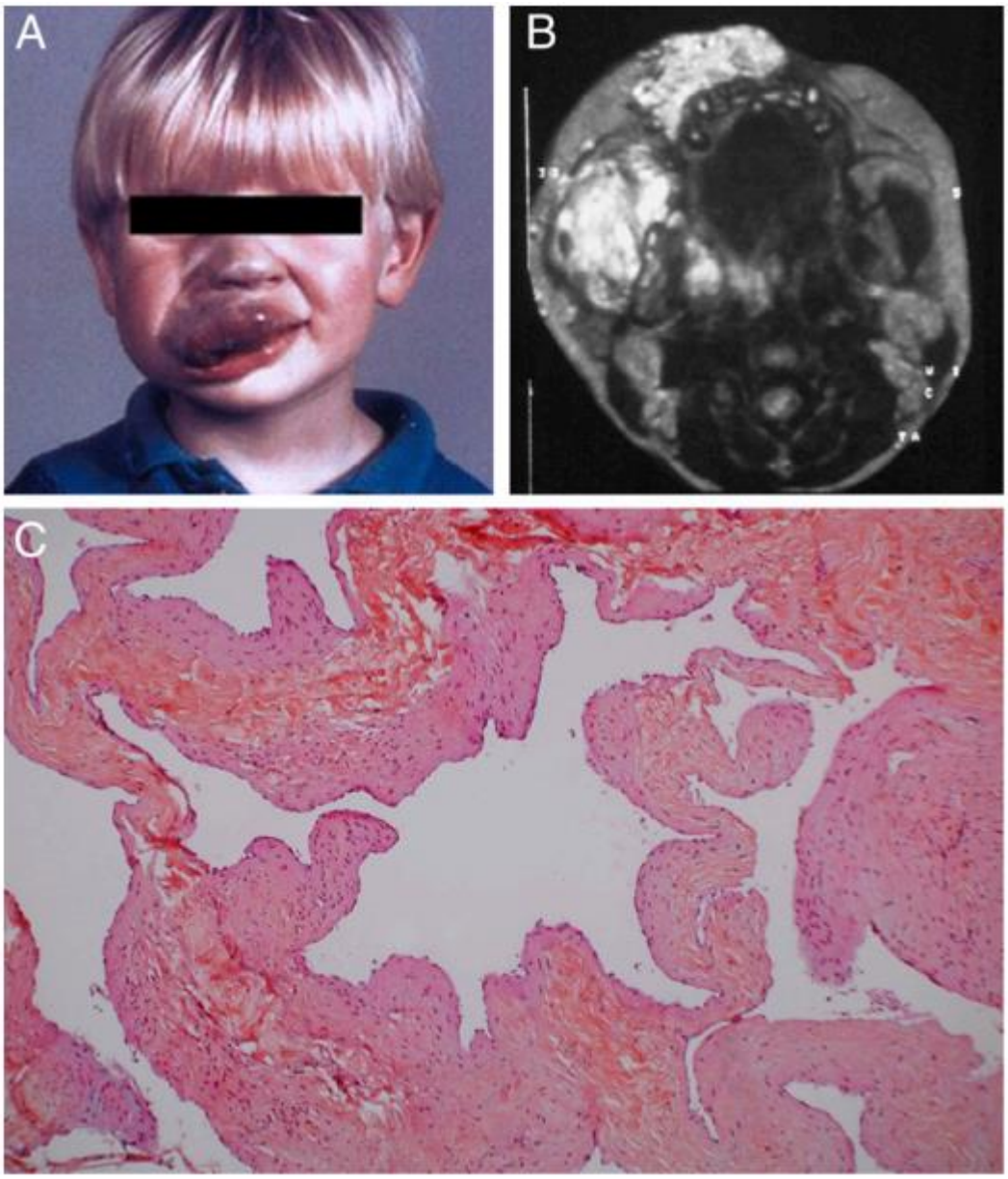


Figura 1.8. A, malformación venosa azulada alrededor de labios y cuello. B, MRI de la lesión mostrada en A. C, Malformación venosa formada por vasos estrellados de paredes delgadas.

Las *malformaciones linfáticas* (LMs) están formadas por canales linfáticos dilatados o quistes rodeados por endotelio linfático. Aparecen en regiones ricas en este tejido como cabeza, cuello axilas, cavidades orales, ingle o mediastino. Se dividen en 3 tipos: macroquísticos, microquísticos y combinados [5] [6].

Los LM macroquísticos son masas quísticas grandes y lisas. Estas lesiones suelen ser superficiales, en piel subyacente o tejido subcutáneo, aunque pueden darse en tejidos más profundos. Las malformaciones linfáticas macroquísticas cervicales se pueden ver asociadas a anomalías cromosómicas como los síndromes de Turner, Noonan y Down [6]. En la Figura 1.9 [5] se puede ver un ejemplo de una malformación linfática macroquística.

Las malformaciones linfáticas microquísticas se manifiestan como pequeñas vesículas, translucidas o hemorrágicas, que pueden confundirse con edemas musculosos. Se dan en una gran variedad de tejidos, pero los más comunes son en la piel y superficies mucosas [6].

Otro tipo de malformaciones linfáticas son las malformaciones linfáticas mixtas. Este tipo de malformaciones tienen componentes macroquísticas y microquísticas citadas en los párrafos anteriores [6].

Las malformaciones linfáticas, en general, no requieren de diagnóstico por imagen, aunque si el diagnostico no es claro se recurre a la ayuda de ecografías Doppler en color. Las pruebas con resonancia magnética se pueden usar también además de evaluar la extensión de la malformación [7].

Las *malformaciones arteriovesonas* (AVMs) y las *fistulas arteriovenosas* (AVFs) son una comunicación anormal entre arterias y venas. Las AVMs se suelen describir como nidos o enredos de vasos. Las AVFs son un único conducto directo de alto flujo [6]. Las malformaciones arteriovenosas son el tipo de malformación vascular más agresivas [5]. Están presentes desde el nacimiento, pero es común que no se muestren ni desarrollen síntomas hasta pasada la primera o segunda década de vida [6] [7]. En la Figura 1.11 [5] se muestra una AVM de brazo, en el que las manos se han deformado a causa de la enfermedad.

Como consecuencia de estas conexiones de alto flujo, las malformaciones arteriovenosas producen un murmullo palpable en los tejidos próximos a la lesión que puede apreciarse en una examinación médica [6]. Con la ayuda de la imagen médica se pueden diferenciar las malformaciones arteriovenosas, más peligrosas, del resto de malformaciones vasculares. Se recurre a ecografías de ultrasonidos [6] [7] y a imágenes de resonancia magnética (MRI) [7]. Con estos métodos se identifican zonas de alto flujo sanguíneo en las que se encuentra esta lesión, a la vez que se puede diagnosticar mejor el grado de dicha lesión [7].

En la Figura 1.10 se muestra una malformación vascular de alto flujo en la zona lumbar. La imagen pertenece a un paciente del Hospital Virgen del Rocío

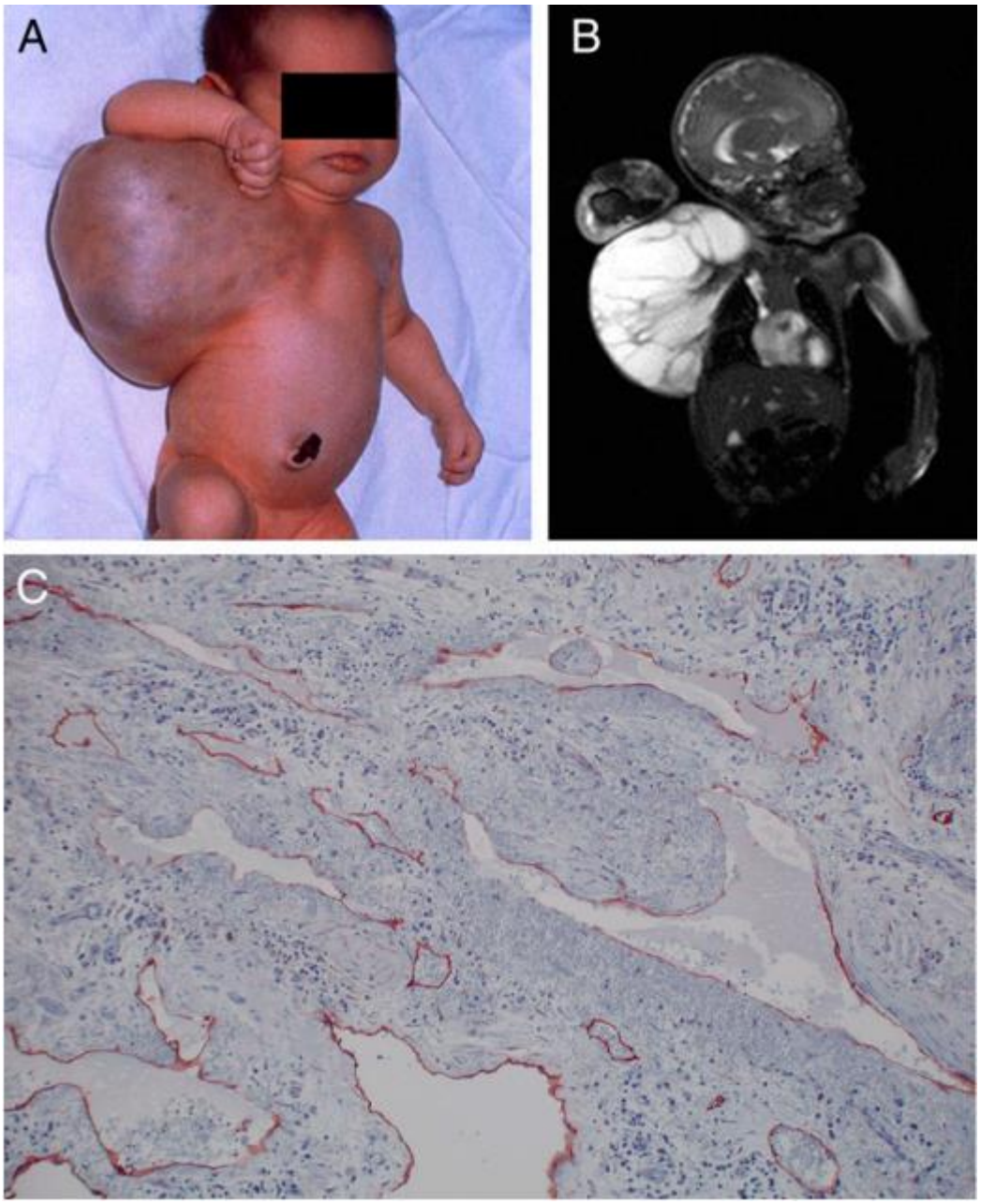


Figura 1.9. A, Gran malformación macroquística en la región de las axilas. B, MRI de la malformación macroquística. C, células endoteliales de un LM.

Figura 1.10. Malformación vascular de alto flujo.

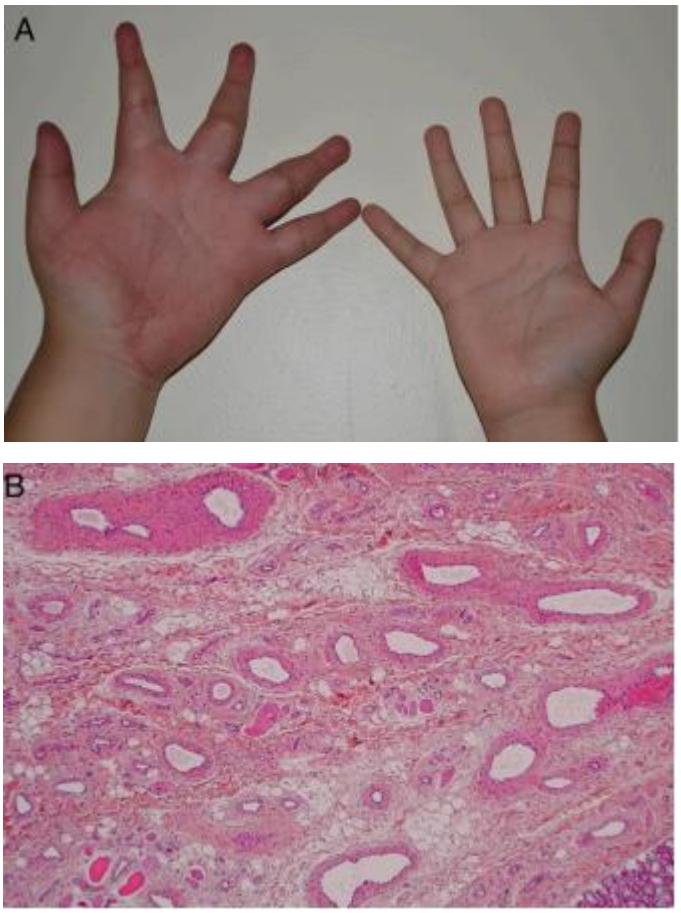


Figura 1.11. Manos alargadas con malformaciones arteriovenosas de brazos. B, Las malformaciones arteriovenosas se forman como óvalos de distintos tamaños, distribuidos por pequeños vasos sanguíneos

- Malformaciones combinadas

Las malformaciones vasculares combinadas combinan 2 o más lesiones vasculares en una única malformación [5]. Las malformaciones vasculares combinadas pueden contener componentes profundos subyacentes a las malformaciones superficiales más evidentes clínicamente y, por lo tanto, estas lesiones pueden requerir evidencia radiológica y / o histopatológica para hacer el diagnóstico correcto [6].

- Malformaciones de vasos sanguíneos principales

Estas malformaciones afectan a venas, arterias o tejidos linfáticos de gran importancia. Las fístulas arteriovenosas congénitas y la persistencia de los vasos embrionarios también están incluidas en este grupo de malformaciones [5]. Las anomalías incluyen anomalías de origen, curso, número, longitud, diámetro (por ejemplo, hipoplasia, ectasia o estenosis) y / o válvulas [6]. Estas malformaciones vasculares se caracterizan por la estabilidad y el potencial de reparación sin que se vuelva a reaparecer la anomalía.

- Malformaciones asociadas a otras anomalías

  Las malformaciones vasculares, simples, combinadas o de vasos sanguíneos principales, pueden estar asociadas con anomalías de huesos, tejidos blandos, o vísceras con un sobrecrecimiento [5] [6]. La mayoría de estas asociaciones son síndromes homónimos. Así por ejemplo la asociación de malformaciones capilares, con venosas, linfáticas y el sobrecrecimiento de las extremidades se conoce como el síndrome de Kippel-Trenaunay.

Tradicionalmente todos estos tipos de malformaciones se han dividido en malformaciones de *bajo flujo* (capilares, venosas, linfáticos o lesiones combinadas) y de *alto flujo* (lesiones con componente arterial). Examinando ambos grupos se puede distinguir las lesiones de alto flujo por su calor y porque al palpar se pude puede apreciar un "murmullo" sobre la zona afectada [6].

Las lesiones de *bajo flujo* últimas suelen estar presentes desde el nacimiento como hemos visto anteriormente, y nunca desaparecen. Cambian su posición, tamaño o histología y son muy difíciles de tratar [1]. En este grupo entran las malformaciones capilares, venosas, linfáticas y malformaciones combinadas sin componente arterial.

Las malformaciones de *alto flujo* son especialmente peligrosas e interesa su diagnóstico y evaluación ya que tienen tendencia a crecer y evolucionar con mayor facilidad que las lesiones de *bajo flujo*. Están presentes desde el nacimiento y crecen con el paso del tiempo con una rápida expansión [1]. Las malformaciones arteriovenosas son el principal representante de este grupo, así como lesiones combinadas con componentes arteriales.

1.4 Objetivo del proyecto

El objetivo de este proyecto, como ya se ha introducido, es la implementación de un sistema de bajo coste que permita diagnosticar y diferenciar entre las malformaciones de bajo y alto flujo. Además, el sistema permite el seguimiento de los hemangiomas, tumores benignos que siempre provocan variaciones térmicas. Debido al comportamiento de ambos grupos de malformaciones, se han podido diferenciar a través de MRI dado su bajo y alto grado de movimiento en los flujos [4]. Esto hace que sean también diferenciables a través de la temperatura: las malformaciones de bajo flujo se encuentran a una temperatura menor que las de alto flujo que incrementa la temperatura corporal en la región afectada [4] [5] [6]. El objetivo de este proyecto es la implementación de un dispositivo de bajo coste, manejable y versátil para un rápido diagnóstico de las malformaciones vasculares a través del análisis de imágenes infrarrojas.

Dentro del proyecto del que forma parte este proyecto, uno de los problemas a solucionar problema de la implementación de los algoritmos para el procesado de la imagen y la creación de una interfaz intuitiva. El objeto de este trabajo es la implementación de estos algoritmos de procesamiento de la imagen en el espectro

infrarrojo para la detección de las malformaciones vasculares y el diseño e implementación de una GUI que facilite el uso de estos algoritmos y sea accesible para cualquier usuario.

1.5 Motivaciones para desarrollar el sistema propuesto

Desde antes de finalizar el Grado en Ingeniería Electrónica Industrial mis intereses fueron enfocados a la electrónica con aplicaciones biomédicas, lo cual, es la principal motivación para realizar este trabajo.

Descubrir cómo usar una Raspberry Pi también es unos de los retos de este trabajo, así como descubrir las aplicaciones y usos de este dispositivo.

Uno de los campos que más evolucionan en la actualidad es el del procesamiento de imágenes debido a la proliferación de sistemas de visión artificial. En este trabajo se tratarán imágenes médicas para la extracción de información relevante

Como ultima motivación de este trabajo, se encuentra la de aprender a programar en Python, un lenguaje muy extendido en la actualidad, y valuar su utilidad para uso personal y profesional.

La resultante de todas las motivaciones es este trabajo, un sistema portátil de para la adquisición y procesado de imágenes médicas, enfocado en el diagnóstico de malformaciones vasculares.

2 Descripción del sistema

El sistema se pensó a partir de la idea de desarrollar un dispositivo de bajo coste capaz de realizar las funciones que se describen en la sección anterior. Además de un coste reducido se busca un sistema que pudiera usarse en la clínica. Para esto, debe ser portable, autónomo y fácil de usar por personas no expertas en tecnología. Con estos requisitos se piensa en la plataforma Raspberry Pi como dispositivo principal.

La plataforma Raspberry Pi es un ordenador de bajo coste, de tamaño reducido y muy popular a la que se le pueden añadir una gran variedad de periféricos. Eso lo hace ideal para diseñar un dispositivo portátil y versátil, con un sistema operativo sencillo y con aplicaciones que facilitan la implementación de los propios algoritmos. A esta plataforma se le añaden los dispositivos necesarios para el propósito del proyecto.

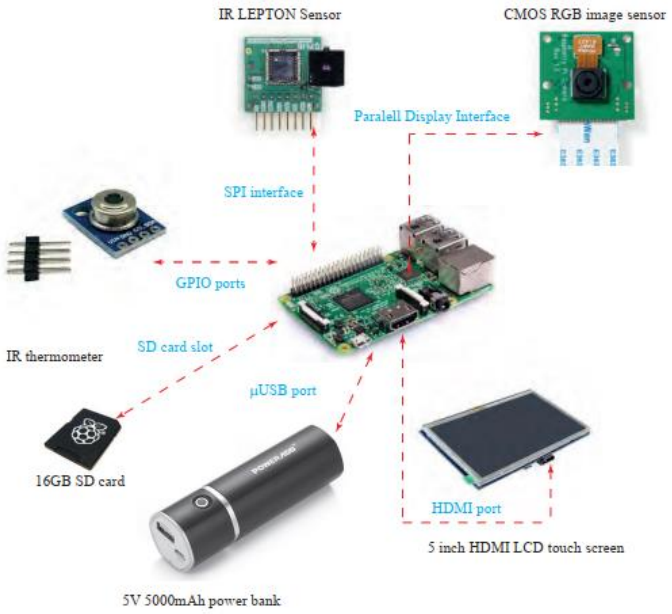


Figura 2.1. Componentes del sistema.

En la Figura 2.1 [10] se muestra un diagrama de la arquitectura del sistema implementado en el proyecto. La descripción detallada se puede encontrar en el artículo "A customizable thermographic imaging system for medical image acquisition and processing" [10]. A continuación, se describen las características de los dispositivos utilizados.

2.1 Raspberry pi

El principal elemento del sistema es la Raspberry Pi. Raspberry Pi es un ordenador completo, simple, de bajo coste y tamaño. En una única placa se incluye CPU, GPU, memoria RAM, bluetooth, WiFi, Ethernet, puerto GPIO entre otros. En la Tabla 2.1 se resumen las características [11] de la Raspberry Pi 3B utilizada en el proyecto.

SOC	Broadcom BCM2837
CPU	4× ARM Cortex-A53, 1.2 GHz
GPU	Broadcom VideoCore IV
Almacenamiento	Micro SD slot
RAM	1GB
Conexión de red	10/100 Ethernet, 2.4 GHz 802.11n wireless
Bluetooth	Bluetooth 4.1 Classic, Bluetooth Low Energy (BLE)
GPIO	40 pines
Puertos	• HDMI • Jack 3.5mm para audio-video • 4x USB 2.0 • Ethernet • Camera serial interface (CSI) • Display Serial interface (DSI)
Dimensiones	87.1 mm x 58.5 mm x 14.8 mm [12]
Sistema operativo	Raspbian OS 10

Tabla 2.1. Características Raspberry PI 3B

Como se puede ver en la tabla, la placa cuenta con un microprocesador de 4 núcleos ARM Cortex-A53 a 1.2 GHz, 1 GB de memoria RAM, la memoria del dispositivo viene dada por la tarjeta micro SD que se le incorpora, Bluetooth 4.1 y BLE, 40 pines de entrada y salida, USB, HDMI, etc. Estas características hacen que sean un dispositivo muy versátil

a la vez que económico que en la actualidad se utiliza en infinidad de fines, muchas de ellas orientadas a IoT aprovechando su conectividad. La potencia del microprocesador que monta la Raspberry Pi junto con los puertos GPIO, el puerto CSI, específico para comunicación con cámaras, y la posibilidad de mostrar la información por pantalla a través del puerto HDMI hacen posible el procesamiento de imágenes en tiempo real. En la Figura 2.2 se muestran distintas perspectivas de la placa la Raspberry Pi 3B.

El sistema operativo utilizado es Raspbian OS 10, un sistema operativo Linux, basado en Debian para Raspberry Pi. Bajo este sistema operativo, el sistema se desarrolla en Python 3. El sistema operativo dispone de aplicaciones hacen más amigable la programación en este lenguaje como es el caso de ThonnyPy, el IDE utilizado para desarrollar la interfaz en este trabajo.

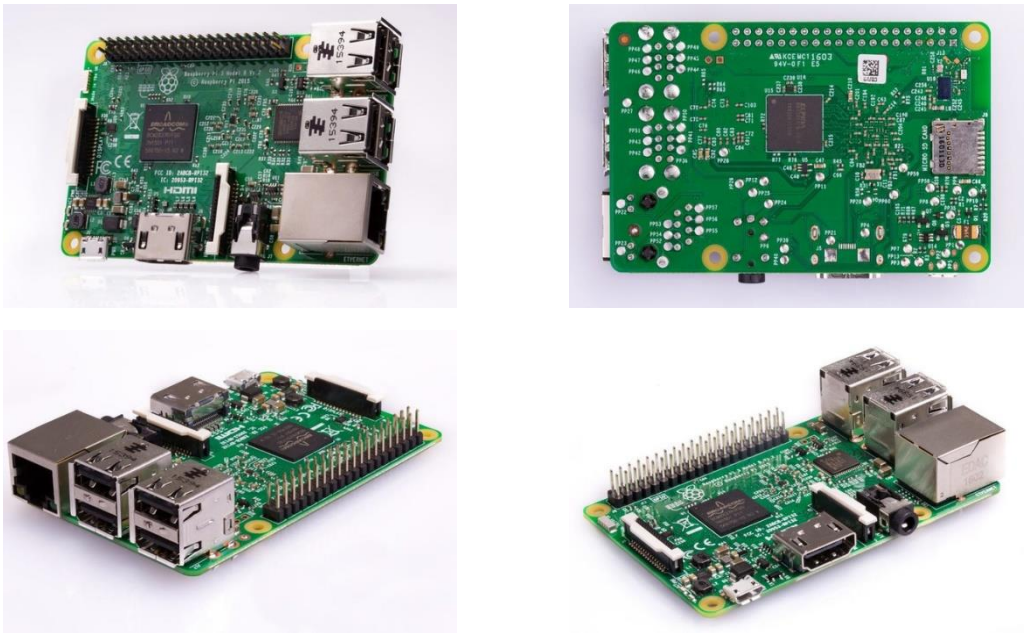


Figura 2.2. Imágenes Raspberry Pi 3B.

La Raspberry Pi controla todos los periféricos, procesos, almacenamiento, procesamiento e imágenes y muestra las imágenes por la pantalla.

2.2 Sensor infrarrojo

Como sensor infrarrojo se elige el FLIR Lepton 2.0 [14]. Se trata de un sensor LWIR (Long wave infrarred) de tamaño y consumo de potencia reducido y de bajo coste. Proporciona una imagen termográfica con una resolución de 80 x 60 pixeles con una tasa de refresco de 9 frames por segundo. En la Tabla 2.2 se resumen las características [15] del sensor.

Resolución de la imagen	80 x 60 pixeles
Frame rate	9 Hz – 30 Hz
Ángulo de visión horizontal	Hasta 50” dependiendo de la variante
Dimensiones	8.5 mm x 11.7 mm x 5.6 mm
Interfaces	• I2C para control • SPI para transmisión de imagen
Alimentación	• Core: 1.2V • Sensor: 2.8 • I/O: 2.5V – 3.1 V
Consumo de potencia	150 mW

Tabla 2.2 Características Lepton 2.x.

Al ser un dispositivo de bajo consumo de potencia y de tamaño reducido facilita la implementación del dispositivo como un dispositivo portable. En la Figura 2.3 [26] se muestra una imagen del sensor.

Figura 2.3. Sensor infrarrojo Lepton 2.x.

2.3 Otros dispositivos

Como ya hemos mencionado, el objetivo de este trabajo fin de máster es la implementación de algoritmos para el procesado de imágenes infrarrojas además de la creación de la interfaz de usuario para el uso de los mismo. Es por esto que los dispositivos de mayor interés son la Raspberry Pi y el sensor IR descrito en las secciones anteriores pero el proyecto consta de más dispositivos para cumplir con el objetivo final. Los periféricos son los siguientes

- **Sensor de imagen RGB**: Para adquirir imágenes en el espectro visible se dispone de un sensor de imagen RGB. Estas imágenes ayudan a comparar las imágenes infrarrojas con la imagen visible de las anomalías adquiridas. Proporcionan información útil para el análisis como la textura o el color [10].

- **Termómetro IR:** Para calibrar el sensor infrarrojo es necesario medir la temperatura en el cuerpo. Con el termómetro IR se adquieren medidas de

temperaturas absolutas sin necesidad de entrar en contacto con el paciente [10].

- **Pantalla LCD:** El dispositivo usa una pantalla táctil de 5 pulgadas de tipo LCD. En ella se muestran las imágenes infrarrojas en tiempo real y permite la interacción con el sistema [10].

- **Batería externa:** El sistema está pensado como un sistema portátil. Por esto es necesario una batería que alimente el dispositivo. Se hace uso de una batería externa de lítico con una autonomía de 5.5h [10].

- **Tarjeta de memoria externa:** Para instalar el sistema operativo y ejecutar el programa se usa una tarjeta de memoria micro SD de 16GB [10].

2.4 Interfaces del sistema

En esta sección se definen las interfaces entre los elementos del sistema.

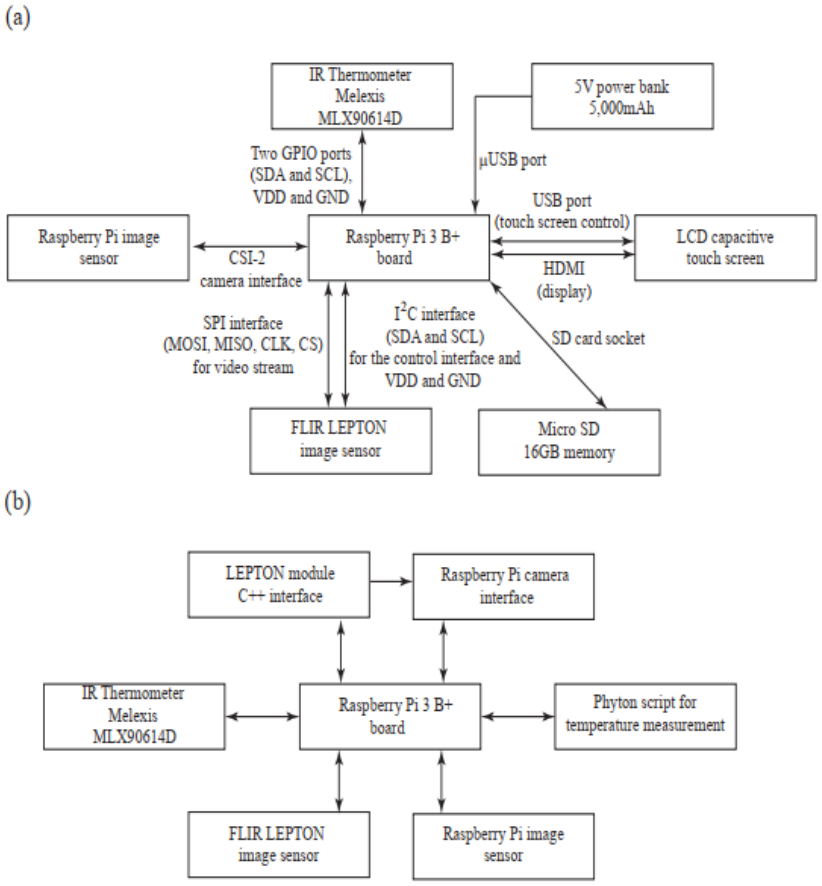


Figura 2.4. (a) Diagrama de interconexión (b) esquema de las interfaces del sistema.

En la Figura 2.4 se muestra la conexión de la Raspberry Pi con los distintos periféricos que componen el sistema.

2.4.1 Raspberry PI – Sensor IR

Como se muestra en la Figura 2.4 se necesita establecer 2 protocoles de comunicación entre la Raspberry Pi y el sensor:

1. SPI: El sensor envía los frames a través del bus SPI
2. I^2C: Se utiliza para enviar los comandos de control al sensor.

- BUS SPI [25]

El bus SPI sigue una configuración maestro-esclavo y de tipo síncrono compuesto por 4 hilos:

- SCLK: Señal de reloj generada por el maestro. Se utiliza para sincronizar la transferencia de datos.
- MOSI (*Master Output Slave Input*): Hilo para la comunicación de maestro a esclavos. Todos los esclavos se tienen que conectar a este hilo para que puedan ver el mensaje del maestro.
- MISO (*Master Input Slave Output*): Hilo para la comunicación de los esclavos al maestro
- CS (*Chip Selector*): A través de este hilo el maestro selecciona el esclavo con el que se quiere comunicar. Hay una salida en el maestro para cada uno de los esclavos.

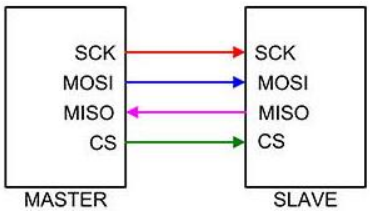


Figura 2.5. hilos de comunicación del bus SPI con 1 maestro y 1 esclavo.

En la Figura 2.5 se muestra el diagrama de conexión entre un maestro y un esclavo para el bus SPI. Nótese que la línea CS tiene que repetirse por cada uno de los esclavos que se encuentran en el bus. Para la transmisión de la información son necesarios los siguientes pasos:

1. Para iniciar la comunicación el maestro pone en estado bajo únicamente la salida CS del esclavo con el que desea comunicarse.
2. El maestro inicial la señal de reloj SCLK de acuerdo a la velocidad de transmisión.
3. En cada ciclo de reloj el maestro escribe un bit en el hilo MOSI y lee un bit del hilo MISO. El esclavo realiza el mismo trabajo leyendo de la línea MOSI y escribiendo en la MISO.

4. Cono finalizan las comunicaciones el maestro deshabilita la señal reloj y pone la línea de CS en alto, indicando que la transferencia de información con el esclavo ha acabado.

En la Figura 2.6 se muestra un ejemplo de la comunicación SPI ilustrando los paso descritos.

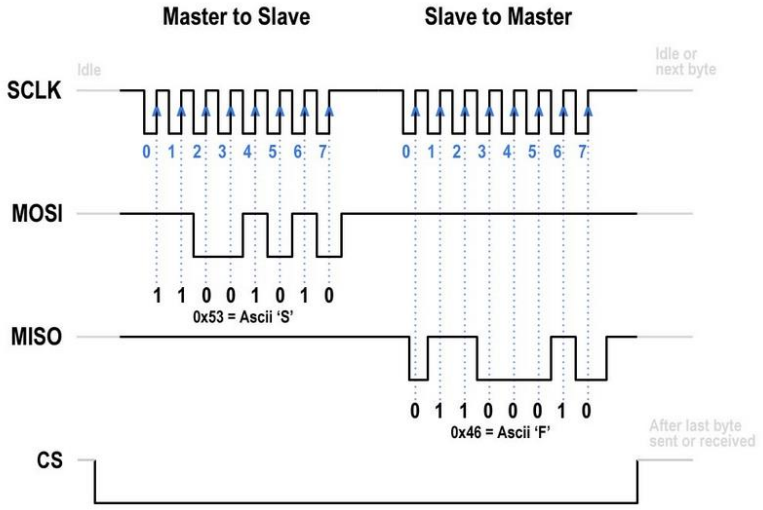


Figura 2.6. Ejemplo de comunicación maestro-esclavo en bus SPI.

- BUS I2C [25]

El bus I2C se trata de otro bus de comunicación síncrono con configuración maestro-esclavo. En este caso el bus cuenta con 2 hilos:

- SCL: Señal de reloj.
- SDA: Bus de datos.

Los dispositivos se pueden configurar como maestros o esclavos en el bus. La señal de reloj la controla siempre el maestro y los esclavos solo contestan a peticiones del maestro.

Los menajes del bus I2C se estructuran de acuerdo a la Figura 2.7 [25].

Mensaje								
Start	7 o 10 Bits	Bit para Leer/ escribir	Bit para reconocer ACK/ NACK	8 Bits	Bit para reconocer ACK/ NACK	8 Bits	Bit para reconocer ACK/ NACK	Stop
Condición de inicio	Dirección			Transporte información		Transporte información		Condición de paro

Figura 2.7. Estructura mensaje I2C.

Durante la comunicación los bits del SDA solo cambian cuando la señal SCL está en estado bajo. En el único momento en el que la señal SDA cambia a nivel bajo antes que la señal de reloj es para iniciar la comunicación.

Los primeros 7 o 10 bits corresponde a la dirección del esclavo con el que el maestro quiere comunicarse. Los datos se mandan a todos los dispositivos que hay en el bus, pero solo responde el que ha sido preguntado a través de la dirección.

El siguiente bit (R/W) indica si se realiza un mensaje de lectura o de escritura. En el mensaje de escritura el maestro manda información al esclavo indicado y en la operación de lectura el esclavo es preguntado.

Con el bit de ACK (Acknoledge) el esclavo confirma la recepción del mensaje.

Dependiendo de la cantidad de información que se quiera enviar se realiza esta operación es o menos veces. Al final de la transmisión se envía la condición de parada, poniendo el hilo SDA en alto cuando el reloj está en estado alto.

En este proyecto, en ambas comunicaciones, el maestro es la Raspberry Pi y el esclavo es el sensor IR.

2.4.2 Raspberry PI - Display

Para mostrar la imagen en una pantalla se recurre al HDMI nativo de Raspberry PI, pudiéndose mostrar la imagen en cualquier display con entrada HDMI.

Al tratarse de una pantalla táctil también es necesaria la comunicación USB con la Raspberry Pi.

2.4.3 Raspberry Pi - Termómetro IR

Al igual que el control del sensor IR, la temperatura del termómetro se transmite a través del bus I2C

2.4.4 Raspberry Pi - Sensor RGB

La comunicación entre el sensor RGB y la Raspberry se establece a través del bu serie específico para cámaras CSI-2

El dispositivo esta pensado como un sispositivo portatil, de facil transporte. Para el proyecto se diseño una carcasa con un apendice para la sujección manual como se muestra en la Figura 2.8 [10]. En la Figura 2.9 se uestran detalles de la implementación final del dispositivo [10].

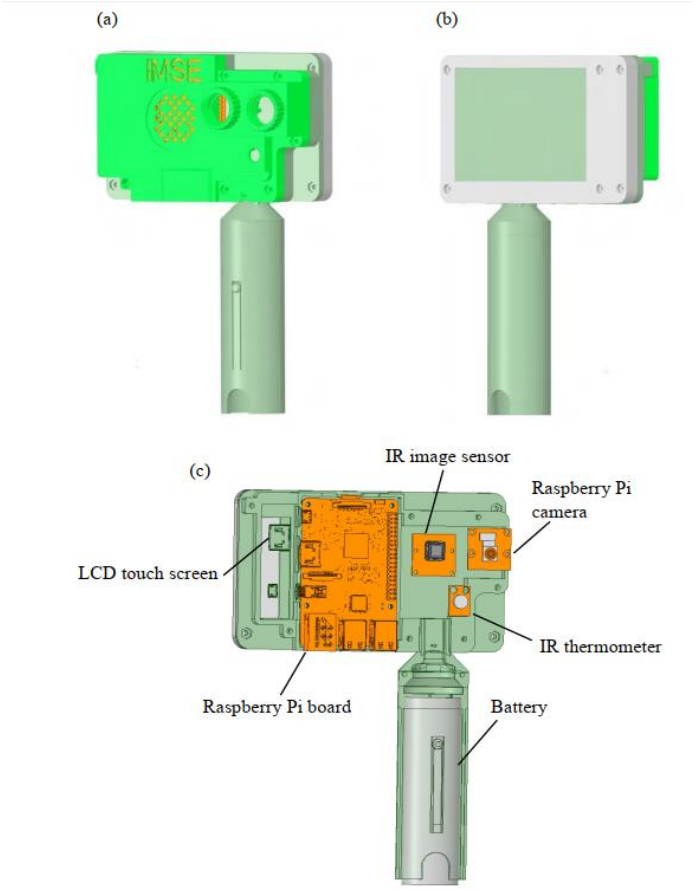


Figura 2.8. (a) Vista trasera, (b) vista frontal (c) diseño interno mostrando la isposiciónde los componentes.

La carcasa se fabricó con una impresora 3D siguiendo los siguientes requisitos [10].

1. Dispositivode fácil mnejo durante las examinaciones medicas.
2. La bateria es facilmente reemplazable.
3. Ligero
4. Conexiones e la RaspberryPi accesibles
5. Debe permitir la refrigeración de la Raspberry Pi.

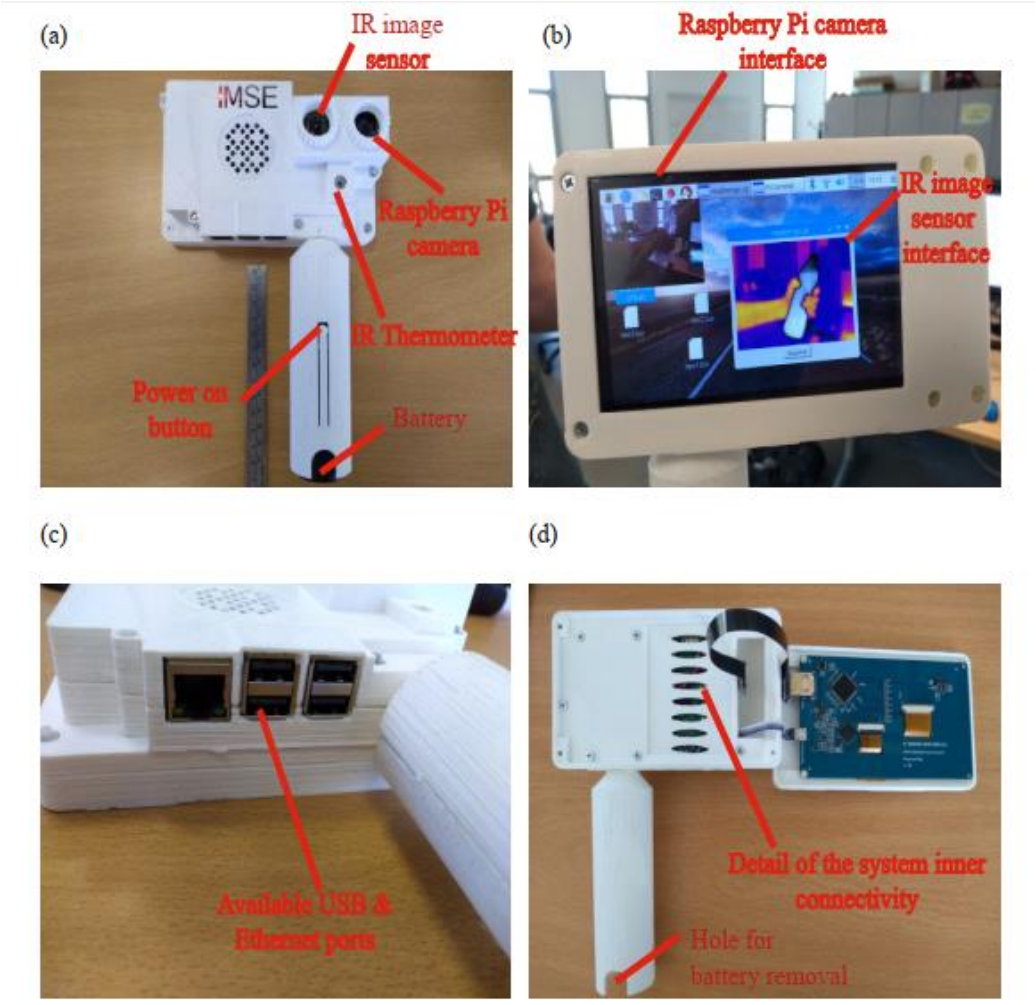


Figura 2.9. Imágenes de la implementación final.

Las cámaras se colocan juntas para obtener la imagen desde la misma posición y se configuran para obtener las imágenes al mismo tiempo.

El pesio total del dispositivo es de 850g incluyendo la batería. Cuenta con una batería de 890mAh lo que da una autonomía de unas 5.5h [10].

3 Funciones de procesado e interfaz de usuario

El sensor IR proporciona datos relacionados con el nivel de radiación infrarroja que capta. Esta información no es útil para el diagnóstico. Con este motivo en el trabajo se desarrollan algoritmos para transformar esta información en temperatura, información que si es útil para el diagnóstico. En el dispositivo inicial únicamente se contaba con una rutina en C++ que representaba la imagen termográfica. Para realizar un diagnóstico rápido y en situ es necesario obtener alguna información añadida en tiempo real como es el caso del gradiente de temperatura y las líneas isotermas que permiten identificar rápidamente las variaciones de temperatura generadas por las anomalías vasculares.

Para que estos algoritmos los pueda utilizar cualquier persona que no esté familiarizada con el código y hacer su uso más amigable es necesario la implementación de una interfaz de usuario o GUI. A través de una ventana y unos simples botones cualquier persona puede hacer uso de estos algoritmos.

Estos algoritmos se implementan en lenguaje Python, unos de los más extendidos en la actualidad. A parte de ser compatible con la plataforma Raspberry Pi, este lenguaje se elige este lenguaje por lo fácil e intuitivo que es además de la cantidad de librerías para casi cualquier propósito que ay disponibles. Esto facilita la implementación y optimización de los algoritmos y la interfaz del trabajo. En otros lenguajes como C++ esto sería mucho más laborioso y enrevesado.

En este capítulo vamos a desarrollar paso a paso el código y su fundamento teórico. En la Figura 3.1 se muestra el diagrama de flujo simplificado. A continuación, se describen cada una de las funciones.

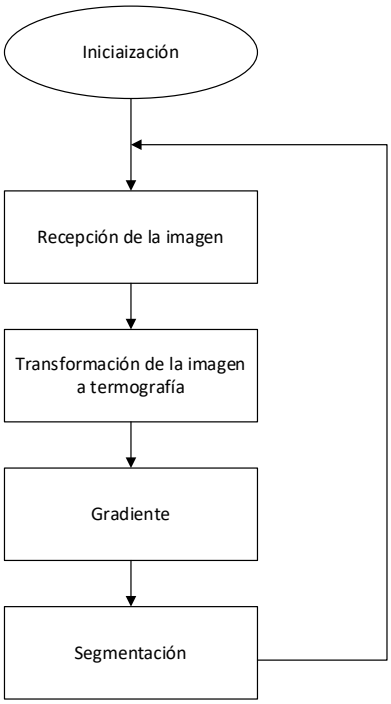


Figura 3.1. Diagrama de flujo del programa.

3.1 Recepción de la imagen

El primer paso es recibir la imagen de la cámara infrarroja. La forma en la que recibimos la imagen de la cámara es un archivo ".bin" con 9604 bytes de información. Se trata de 4802 elementos en formato "unsigned long". Estos 4802 datos corresponden a:

- La primera palabra es el valor mínimo de la escena.
- La primera palabra es el valor máximo de la escena.
- El resto corresponden a los 4800 valores de intensidad de una imagen de 60*80 pixeles.

El archivo ".bin" se desempaqueta con la función "struct.iter_unpack(format, buffer)" [13] en una lista de 4802 elementos. Inmediatamente se transforma la lista en un array unidimensional con el mismo número de elementos.

Después de esto se forma la imagen con los respectivos 60x80 pixeles. Para esto se trata se almacenan en un array de 60 filas y 80 columnas las 60 filas de 80 pixeles. En un principio se colocaba cada pixel individualmente en el array 60x80 lo que incrementaba el tiempo de procesado de la imagen. En la Figura 3.2 se muestra la imagen obtenida con la cámara FLIR de la espalda de un bebe. En eta imagen se puede ver como los niveles de intensidad proporcionados por la cámara van desde 7900 a 8300.

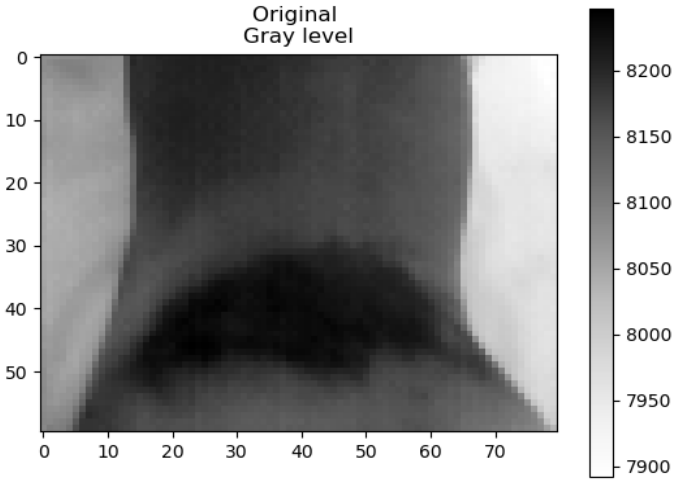


Figura 3.2. Imagen infrarroja raw125.bin

Para ello las funciones utilizadas son OpenFile y List2Im descritas en el Anexo II.

3.2 Transformar a termografía

Como hemos visto en el apartado 3.1, los datos proporcionados por la cámara son poco intuitivos. Para entender mejor estos valores es necesario transformar los valores actuales a valores de temperatura. Para este objetivo el sistema cuenta con un termómetro IR descrito en la sección 2.3. La calibración de este dispositivo (obtener los datos de temperatura para convertir los niveles de radiación infrarroja en temperatura) no es objeto de este trabajo y en todo momento se consideran las medidas de temperatura como una entrada al sistema. Se asume que la calibración es correcta y se consideran 2 puntos medidos experimentalmente con dicho sensor para la implementación del sistema.

Existe una dependencia lineal entre la entre las salidas del sensor que representan radiación en la banda del infrarrojo lejano (LWIR) y la temperatura [25]. El procedimiento que se sigue para la calibración del sensor consiste en medir la temperatura en 2 punto de los que conocemos el valor correspondiente a la salida del sensor IR. A haber una dependencia lineal se puede considerar que la calibración sigue la siguiente expresión:

$$T = IR \cdot a + b \qquad \textit{Ecuación (3.1)}$$

Donde T es el valor del temperatura del pixel (x, y), IR es el valor que proporciona el sensor para el pixel (x, y) y a y b son coeficientes correspondientes a la calibración. Estos coeficientes se obtienen al conocer 2 parejas de puntos $(T_i,\ IR_i)$. Con la misma Ecuación (3.1) se puede obtener el valor de dichos coeficientes. Una vez obtenido el

valor de a y b se aplica la relación lineal a toda la matriz que compone la imagen, obteniendo un valor de temperatura para cada uno de los píxeles [25].

Para obtener los valores de temperatura absolutos se midió la temperatura, con el termómetro IR descrito en la sección 2.3, en 2 puntos T1 y T2 obteniéndose una temperatura en el primero de 36.6 ºC y 36.8 ºC en el segundo. Conociendo el valor de intensidad captado por la cámara en esos puntos R1 = 8250 y R2 = 8300, se puede hacer una aproximación lineal para el resto de pixeles de la imagen. Este proceso se realiza en la función Im2Term descrita en el Anexo II. El resultado se puede ver en la Figura 3.3. Para visualizar de forma más clara la imagen se aplica un mapa de colores.

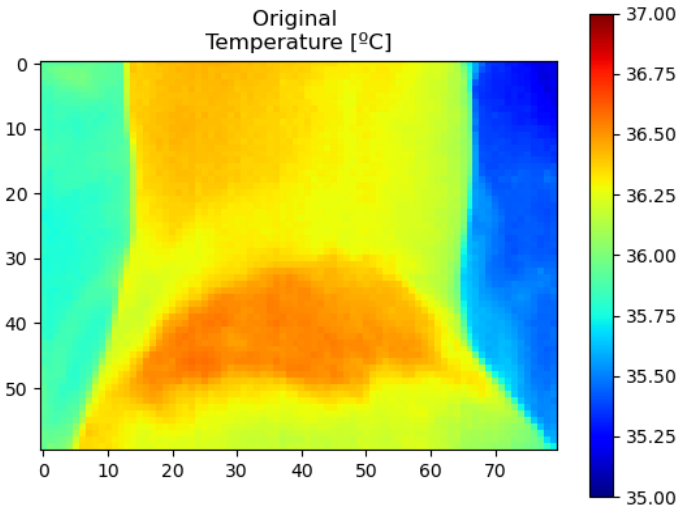


Figura 3.3. Imagen térmica obtenida a partir del archivo raw125.bin.

En esta imagen se puede apreciar la espalda de un bebe. La temperatura corporal se encuentra a uno 36.25ºC. En la zona lumbar se puede apreciar una mancha con una temperatura algo superior a la del resto del cuerpo.

3.3 Detección de bordes

La detección de bordes juega un papel importante a la hora de evaluar las malformaciones vasculares y delimitar su extensión. Con la detección de bordes se puede identificar fácilmente la transición entre una zona con alta temperatura (zona de alto flujo y por tanto de los hemangiomas) con el resto del área no afectada. En este apartado se aplicará un filtro espacial a la imagen con las temperaturas para obtener la variación de las mismas. Esta operación hace uso de lo que se denomina máscara, filtro o kernel. Esta máscara está compuesta de coeficientes que se aplican al entorno del pixel de interés. El proceso consiste en mover la máscara pixel a pixel por la imagen. Para cada punto (x, y) se calcula la respuesta del filtro de una manera concreta. Se conoce como filtro lineal a aquel filtro cuya salida es una combinación lineal de cada una

de las entradas. La salida del filtro lineal se obtiene con la siguiente expresión cuando se aplica una máscara de tamaño m x n:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} z(s,t) f(x+s, y+t) \qquad \textit{Ecuación (3.2)}$$

Donde $f(x,y)$ es la imagen que se desea filtrar, $z(s,t)$ son los coeficientes del kernel. $a = (m-1)/2$ y $b = (n-1)/2$. a y b son enteros no negativos.

Cuando se utiliza un kernel en el que $m \neq n$ se está dando distinta importancia a los vecinos horizontales y verticales del pixel que se está procesando. Lo más común es utilizar kernel cuadrados con $m = n$. El tamaño más usado para los kernels en filtrado espacial es $m = n = 3$ como el que se muestra en la Figura 3.4. Estos kernel utilizan la información de 8 pixeles vecinos al pixel (x,y) que se está procesando en ese momento. Este tamaño es suficiente y asequible para un procesado en tiempo real por lo los microprocesadores actuales [16].

z(-1,-1)	z(-1,0)	z(-1,1)
z(0,-1)	z(0,0)	z(0,1)
z(1,-1)	z(1,0)	z(1,1)

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Figura 3.4. Kernel 3x3 genérico (a) nomenclatura tradicional (b) nomenclatura abreviada.

La respuesta G del filtro de puede expresar también siguiéndola nomenclatura abreviada mostrada en la Figura 3.4.

$$G = z_1 f_1 + z_2 f_2 + \cdots + z_{mn} f_{mn} = \sum_{i=1}^{mn} z_i f_i \qquad \textit{Ecuación (3.3)}$$

Donde f_i es el pixel de la ventana de la imagen $f(x,y)$ a la que se le aplica el filtro. Para el caso concreto de $m = n = 3$.

$$G = z_1 f_1 + z_2 f_2 + \cdots + z_9 f_9 \sum_{i=1}^{9} z_i f_i \qquad \textit{Ecuación (3.4)}$$

La Ecuación (3.3) recuerda a la convolución usada en los filtrados en el dominio de la frecuencia. Por este motivo se suele hablar de una convolución de una máscara o kernel con una imagen. La máscara recorre la imagen, aplicándose la convolución a ventanas de la imagen de tamaño m x n. El operador se aplica localmente en ventanas de la imagen del tamaño de la máscara usada centrado en el pixel que se desea procesar en ese momento. En la Figura 3.5 se muestra un ejemplo ilustrativo de como se hace una convolución de una imagen de tamaño M x N con un kernel 3 x 3. Para el pixel de salida $G(1,1)$ parte del pixel $f(1,1)$ de la imagen inicial junto con sus pixeles vecinos. A esta matriz 3 x 3 de la imagen primitiva se le aplica el kernel $z(s,t)$ de la forma descrita en las ecuaciones Ecuación (3.3) y Ecuación (3.4). Una vez realiza la operación para pixel $f(1,1)$ se obtiene el pixel de la imagen filtrada correspondiente $G(1,1)$. El proceso se realiza para cada uno de los pixeles de la imagen $f(x,y)$.

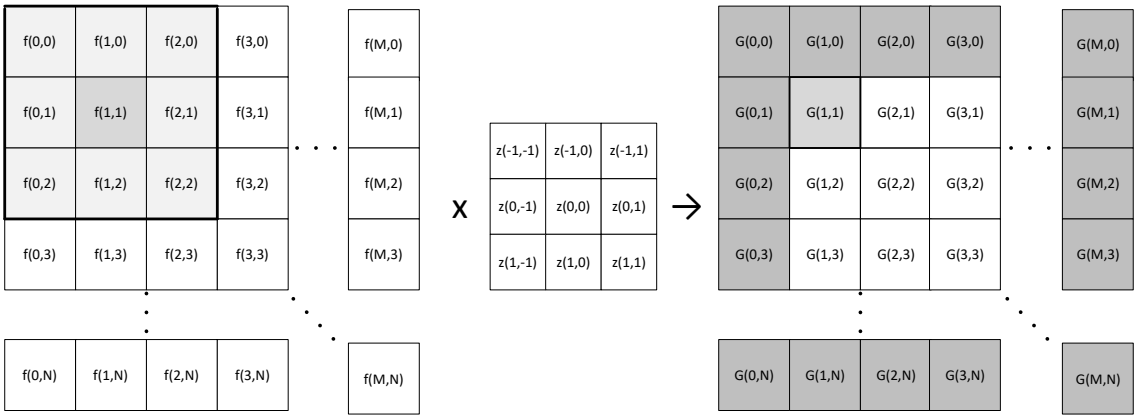


Figura 3.5. Ejemplo ilustrativo convolución con kernel 3x3

Como se puede apreciar en el ejemplo, los bordes de la imagen son un punto conflictivo para el filtrado. Cuando la máscara se mueve cerca de los bordes no hay suficientes pixeles para realizar el filtrado. Hay varias soluciones para este problema. La más simple es limitar el filtrado para que el centro de la máscara no se coloque en los pixeles a una distancia menor de $(n-1)/2$ del borde, considerando una máscara de tamaño n x n. Con esto se obtiene una imagen te menor tamaño que la imagen original, pero todos sus pixeles se obtienen con la máscara deseada. Cuando se requiere que la imagen tenga el mismo tamaño que la imagen original normalmente se filtran los pixeles del borde solo aplicando los componentes de la máscara que se encuentran dentro de los límites de la imagen. Los pixeles de los bordes de la imagen filtrada, en este caso, se obtienen como resultado de un filtro parcial. Otra solución, llamada "padding" o relleno añade filas y columnas de valor "0" o de un nivel constante de gris a la imagen original. Al filtrar esta imagen se obtiene una imagen del mismo tamaño que la imagen original, pero con la influencia en los bordes de estos pixeles de valor constante [16].

En nuestro caso el filtrado de la imagen se realiza para detectar los bordes. Con esto se localizan los puntos en los que hay un mayor cambio de temperatura para realzar las zonas afectadas por las malformaciones vasculares. Para esto se hace uso de la primera

derivada utilizando la magnitud del gradiente. Para una función $f(x, y)$, el gradiente de f en la coordenada (x, y) se define como:

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{df}{dx} \\ \frac{df}{dy} \end{bmatrix} \qquad \textit{Ecuación (3.5)}$$

La magnitud del gradiente viene dada por:

$$mag(\nabla f) = \left[{G_x}^2 + {G_y}^2\right]^{1/2} = \left[\left(\frac{df}{dx}\right)^2 + \left(\frac{df}{dy}\right)^2\right]^{1/2} = \boldsymbol{\nabla f'} \qquad \textit{Ecuación (3.6)}$$

La derivada parcial no es invariante en rotación (isotrópico) de una imagen, pero la magnitud si lo es. Por este motivo tradicionalmente cuando se habla de gradiente, se hace referencia a la magnitud del gradiente [16]. A partir de ahora cuando hablemos de gradiente estaremos hablando de la magnitud del mismo.

Para reducir el cálculo computacional necesario para el procesado del gradiente se recurre a aproximarlo usando el valor absoluto en vez del cuadrado y la raíz cuadrada [16].

$$\nabla f' \approx |G_x| + \left|G_y\right| \qquad \textit{Ecuación (3.7)}$$

Con esta aproximación normalmente se pierde la característica invariante con la rotación, pero es más rápida de calcular. De acuerdo a la Ecuación (3.7) se definen una serie de aproximaciones digitales. Usamos la nomenclatura de la Figura 3.4 (b) para nombrar a una región 3 x 3 de una imagen . Por ejemplo el centro de la imagen $f(x, y)$ es el punto z_5 y el punto z_1 es el pixel $f(x-1, y-1)$. La aproximación más simple viene dada por [16]:

$$G_x = (z_8 - z_5) \text{ y } G_y = (z_6 - z_5) \qquad \textit{Ecuación (3.8)}$$

Roberts propuso otra aproximación en 1965 dada por la siguiente expresión usando diferencias cuadradas:

$$G_x = (z_9 - z_5) \text{ y } G_y = (z_8 - z_6) \qquad \textit{Ecuación (3.9)}$$

Si calculamos la magnitud del gradiente obtenemos:

$$\nabla f' = [(z_9 - z_5)^2 + (z_8 - z_6)^2]^{1/2} \qquad \textit{Ecuación (3.10)}$$

La expresión queda más simple si la aproximamos usando el valor absoluto como se indicó en la Ecuación (3.7):

$$\nabla f' \approx |(z_9 - z_5)| + |(z_8 - z_6)| \qquad \textit{Ecuación (3.11)}$$

Las máscaras asociadas a este gradiente cruzado de Roberts se muestran en la Figura 3.6.

-1	0
0	1

0	-1
1	0

Figura 3.6. Máscaras para los gradientes cruzados de Roberts.

Las máscaras de dimensiones 2 x 2 no son fácil de implementar ya que no tienen un centro claro. La siguiente aproximación dada por Prewitt calcula el gradiente en la dirección x como la diferencia entre la primera y tercera fila, y la derivada en el eje y se obtiene con la diferencia entre la tercera y primera columna:

$$G_x = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3) \quad \textit{Ecuación (3.12)}$$

$$G_y = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7) \quad \textit{Ecuación (3.13)}$$

El operador de Prewitt se muestra en la Figura 3.7.

-1	-1	-1
0	0	0
1	1	1

-1	0	-1
-1	0	1
-1	0	1

Figura 3.7. Mascara del operador de Prewitt.

Para suavizar los gradientes dando más peso a los puntos centrales, se añade un factor 2. Esto da como resultado el operador de Sobel:

$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \quad \textit{Ecuación (3.14)}$$

$$G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \quad \textit{Ecuación (3.15)}$$

En la Figura 3.8 se muestra el operador de Sobel o Sobel Feldman:

-1	0	1
-2	0	2
-1	0	1

-1	-2	-1
0	0	0
1	2	1

Figura 3.8. Operador de Sobel.

En los operadores para la detección de borde, la suma de todos los coeficientes es nula, por lo que en un área con un nivel de gris constante el resultado del gradiente es 0.

En nuestro sistema aplicamos el operador Sobel-Feldman con los siguientes coeficientes [9] [10]:

47	0	-47
162	0	-162
47	0	-47

Figura 3.9. Operador Sobel-Feldman utilizado.

Para aplicar el kernel a la imagen se recurre a la función “convolve2d” de la librería “SciPy”. Esta función es mejora mucho el tiempo de procesado. Con un bucle “for” el cálculo de la convolución de la imagen, sin procesar los bordes es de 353.48 ms de media, mientras que con esta función el tiempo se reduce a 48.63 ms de media. En este caso para calcular los bordes de la imagen se establecen condiciones simétricas respecto al borde [18], añadiendo al otro lado del borde un pixel idéntico al del lado simétrico del mismo (Figura 3.10) para obtener una imagen filtrada del mismo tamaño que la imagen original [19].

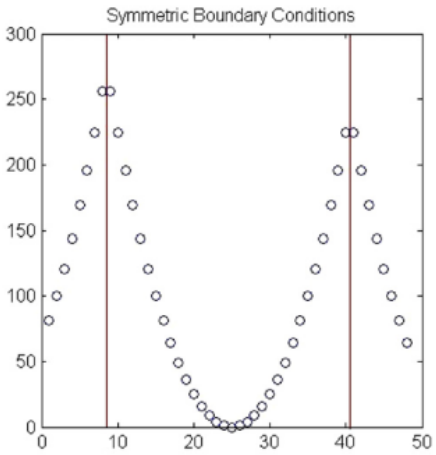


Figura 3.10. Ejemplo de condiciones de borde simétrico.

En la Figura 3.11 se muestra la imagen de Figura 3.3 aplicándole el kernel de la Figura 3.9. En ella se puede ver el gradiente en el eje x de las temperaturas. Se aprecia como en las zonas donde el valor del gradiente es mayor corresponde por un lado al contorno del cuerpo, siendo el límite del cuerpo con la camilla, y por otro, la zona afectada por la malformación.

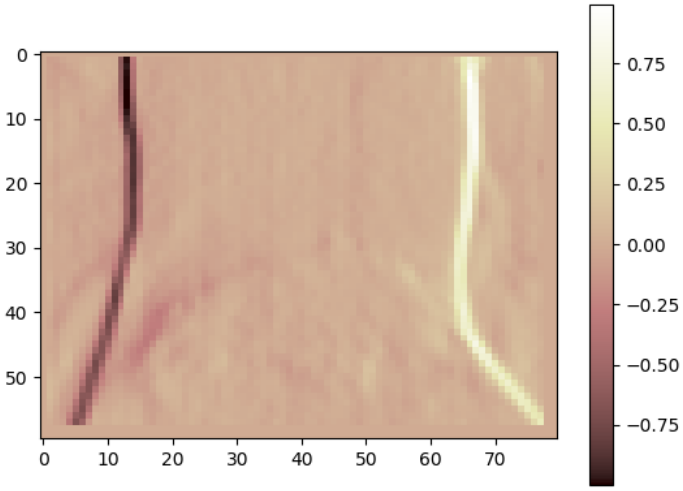


Figura 3.11. Imagen con filtro Sobel aplicado.

3.4 Segmentación

Al proceso en el que una imagen se divide en distintas partes u objetos de la misma se conoce como “segmentación”. La segmentación tiene un papel muy importante en el procesado de imágenes ya que permite extraer la información de interés de la imagen antes de realizar otras tareas como extracción, clasificación, descripción, etc.

El objetivo de la segmentación es dividir la imagen en regiones que podemos etiquetar o diferenciar. Los objetos segmentados se llaman normalmente “primer plano” y al resto de la imagen se le denomina “fondo”. Esta separación entre primer plano y fondo se realiza en función del objeto que se desea aislar.

Para segmentar una imagen se puede hacer mediante 2 métodos: Detección de bordes y por diferenciación de regiones [17].

3.4.1 Detección de bordes

Como definición un borde es un conjunto de pixeles conectados que firman una frontera entre 2 regiones [16]. Idealmente un borde es un conjunto de pixeles conectados de tal forma que se encuentran entre una transición del nivel de gris. En la práctica, los sistemas de adquisición no son ideales y la iluminación, la calidad de la imagen y la frecuencia de muestreo entre otros factores juegan un papel fundamental, ya que añaden ruido a la imagen y hacen que los bodes no sean una simple línea, si no que sea una transición lineal [16].

Para detectar los bordes se hace uso de la primera derivada, descrita en la sección 3.3. También es común el uso de la segunda derivada que nos indica cuando hay un cambio en el gradiente de grises en un borde y menos aún no identifican regiones de una imagen. Con estos métodos no se obtiene un bode como tal, sino que se obtiene información de cómo cambia la imagen. Surgen unos problemas para encontrar los bordes con los gradientes. Los gradientes son muy sensibles al ruido y pueden dar identificar un gradiente donde no hay un borde. También es difícil identificar como de grande tiene que ser un gradiente para que se considere un borde [17].

Normalmente el uso de gradientes va seguido de procesos que enlazan los pixeles de la imagen para así delimitar regiones de la misma [16]. Uno de los procedimientos más simples es el **procesamiento local**. En él se estudian las características de un pequeño vecindario (3 x 3 o 5 x 5) de pixeles de la imagen que se ha identificado como borde a través del uso de la derivada. Las principales características que se usan para identificar si los pixeles pertenecen al mismo borde son la fuerza con la que responden al gradiente y la dirección de vector gradiente [16].

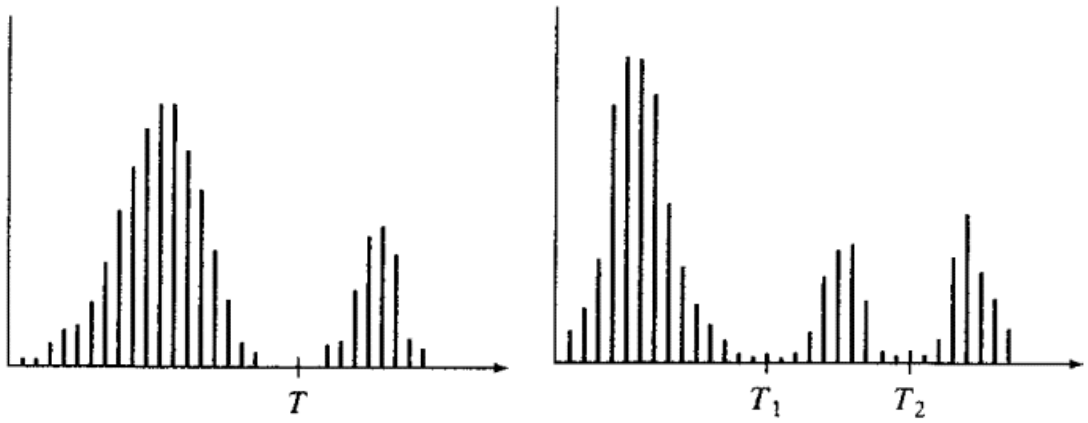
3.4.2 Thresholding

Los métodos de thresholding cada vez son más usados, sobre todo en aplicaciones en las que la velocidad de procesamiento es un factor uy importante [16]. Se basan en establecer un umbral en los niveles de grises que sea capaz dividir la imagen en las regiones deseadas.

Suponiendo un histograma de los niveles de grises de una imagen, $f(x, y)$, compuesta por objetos y un fondo oscuro, los niveles de grises de dicha imagen se pueden agrupar en 2 conjuntos como se puede ver en la Figura 3.12 (a). En este histograma es trivial encontrar un umbral T que separe los 2 bloques con niveles de

grises similares, separando así el fondo de los objetos segmentados. Cuando un punto (x, y) en que se cumple que $f(x, y) > T$ se le denomina "punto de objeto", mitras que si no cumple esta condición ese punto se le llama "punto de fondo".

En la Figura 3.12 (b) se muestra un caso más general en el que se encuentran 3 grupos de niveles de grises dominantes y, por tanto, se pueden diferenciar 2 umbrales T_1 y T_2. De esta forma se clasifica un punto (x, y) como objeto si pertenece a la función $T_1 < f(x, y) < T_2$, a otro objeto si pertenece a la función $f(x, y) > T_2$ o al fondo si $f(x, y) \leq T_1$.



(a) (b)

Figura 3.12. (a) Histograma de los nivele de grises con un umbral destacado y (b) con 2 umbrales.

Se puede ver el este umbral T como una función del punto (x, y), alguna propiedad local $p(x, y)$ de dicho punto y del nivel de gris $f(x, y)$ del mismo

$$T = T[x, y, p(x, y), f(x, y)]$$

Así, se pueden clasificar los pixeles de una imagen por umbral como:

$$g(x, y) = \begin{cases} 1 & si\ f(x, y) > T \\ 0 & si\ f(x, y) \leq T \end{cases}$$

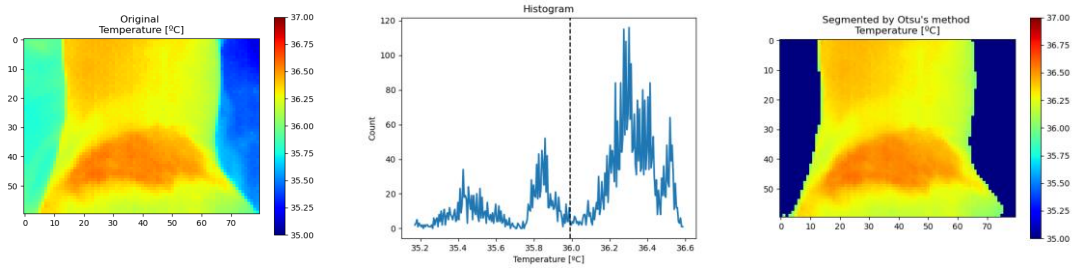
Los puntos etiquetados como '1' corresponden a los objetos, mientras que los 0 corresponden al fondo. Cuando el umbral T depende únicamente del nivel de gris, $f(x, y)$, el umbral se denomina "global". Si el umbral T depende de $f(x, y)$ y de $p(x, y)$ al umbral se le llama "local". Además, si T depende de las coordenadas (x, y) de la imagen se le denomina umbral dinámico o adaptativo.

Para un primer análisis o primera aproximación el umbral se puede elegir visualmente, pero para procesar una secuencia de imágenes es necesario automatizarlo. En la literatura hay diversas formas de hacerlo basadas en el estudio del histograma: métodos iterativos con el nivel de intensidad medio de los pixeles de 2 grupos [16] [1], estudiándose los mínimos de la aproximación polinomial del histograma de la imagen [17][1], o el método de Otsu, que se basa en agrupar los pixeles de manera que la varianza en un mismo grupo sea la menor posible [20] [21].

En el **método de Otsu** se asume que la imagen está compuesta únicamente por el objeto de interés y el fondo. El método divide la imagen en regiones: la región oscura T_0

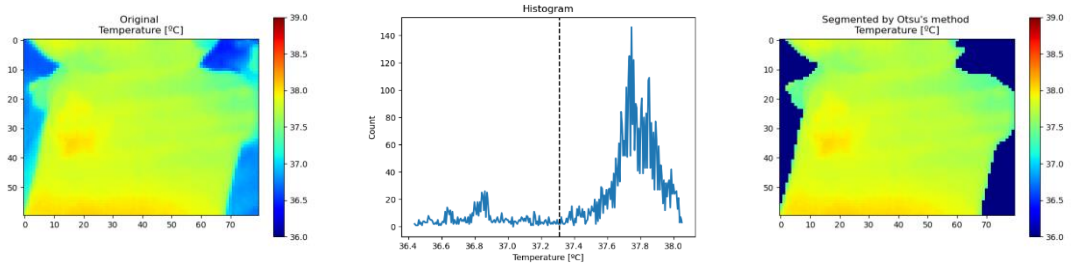
con el conjunto de niveles de intensidad $T_0 = \{0, 1, \dots, T\}$, y la zona clara T_1 con niveles de intensidad $T_1 = \{T,\ T+1, \dots, l-1, l\}$. Siendo T el umbral y l el nivel de intensidad máximo de la imagen. El objetivo es encontrar un umbral tal que la entropía para la suma de los 2 grupos, T_0 y T_1, sea mínima.

Para obtener el umbral a con método de Otsu, hacemos uso de la función "threshold_otsu" de la librería "scikit-image" especializada en procesamiento de imagen [22]. El resultado se muestra en Figura 3.13 y Figura 3.14.



(a) (b) (c)

Figura 3.13. (a) imagen térmica raw125.bin. (b) histograma de la imagen, (c) imagen segmentada.



(a) (b) (c)

Figura 3.14. (a) imagen térmica raw33.bin. (b) histograma de la imagen, (c) imagen segmentada.

En la subimagen (a) se muestra la imagen térmica, en la (b) el histograma de la imagen junto con el umbral obtenido con el método de Otsu y en la (c) la imagen segmentada. Para obtener la imagen segmentada se comparan los niveles de cada uno de los pixeles de la imagen con el umbral obtenido. Si el nivel es más bajo, quiere decir que está a una temperatura menor que la temperatura del umbral y consideramos dicho punto como fondo de la imagen, sustituyendo su valor por "0".

3.5 Cálculo de área

Es interesante monitorizar la extensión de las malformaciones, tanto para diagnostico como para comprobar la evolución durante un tratamiento. Para ello en primer lugar se obtienen M líneas de contorno para un rango de temperaturas en las que se encuentran habitualmente las malformaciones. Una vez representadas las líneas de contorno se elige la temperatura que delimita la anomalía, creando un área de

interés. Para el cálculo del área se usa el método de coordenadas. Se obtienen N pares de coordenadas para dicha región. Para obtener los pares de coordenadas usamos la función "measure.find_contours" de la librería "scikit-image" [22]. El primer y último par de coordenadas son el mismo y se tienen en cuenta 2 veces para el procesado. Primero se calculan los parámetros S_1 y S_2 de acuerdo a la Ecuación (3.16) y Ecuación (3.17):

$$S_1 = x_1 y_2 + \sum_{i=1}^{N-1} x_i y_{i+1} \qquad \textit{Ecuación (3.16)}$$

$$S_2 = x_2 y_1 + \sum_{i=1}^{N-1} x_{i+1} y_i \qquad \textit{Ecuación (3.17)}$$

Una vez obtenidos estos valores el área se calcula según la Ecuación (3.18)

$$Área = \frac{|S_1 - S_2|}{2} \qquad \textit{Ecuación (3.18)}$$

Con esto se obtiene el número de pixeles dentro del contorno. Este método permito obtener una estimación del área en medida absoluta si se obtienen las imágenes a una misma distancia y con la misma óptica [10].

En la sección 4 se muestran varios ejemplos del cálculo del área de distintas zonas afectadas por malformaciones vasculares.

3.6 Diseño de la GUI

El mayor reto del trabajo es tener una carga computacional baja, para conseguir unos tiempos de respuesta razonables. Se ha tratado de reducir el tiempo de ejecución de los algoritmos como se ha descrito en el capítulo anterior. Sin embargo, la mayor carga computacional es debida a representar las imágenes. La herramienta más extendida para representar imágenes y gráficos en Python es "matplotlib". Con esta herramienta los tiempos para representar las imágenes eran superiores a 1s. El sensor infrarrojo es capaz de proporcionar unos 9 fps. Este tiempo que tarda en representar las imágenes hace que no sea factible su uso para esta aplicación.

Para reducir este problema se hace uso del framework Qt a través de su librería para Python pyQt. Qt es un framework que proporciona distintas estructuras y herramientas para el desarrollo de aplicaciones gráficas. En este proyecto usamos una distribución para Python de Qt de código libre bajo licencia LGPL denominada "pyQt". Además de reducir el tiempo para representar una imagen, dispone de los elementos necesarios

para crear un entorno gráfico e intuitivo. También se hace uso del paquete “pyQtGraph”, una librería de propósito científico y para el ámbito de la ingeniería que hace la que computación de matrices, en nuestro caso imágenes, sea más rápida [24].

En la Figura 3.15 se muestra la estructura de la GUI creada en este trabajo. En los siguientes puntos se describen cada uno de los elementos.

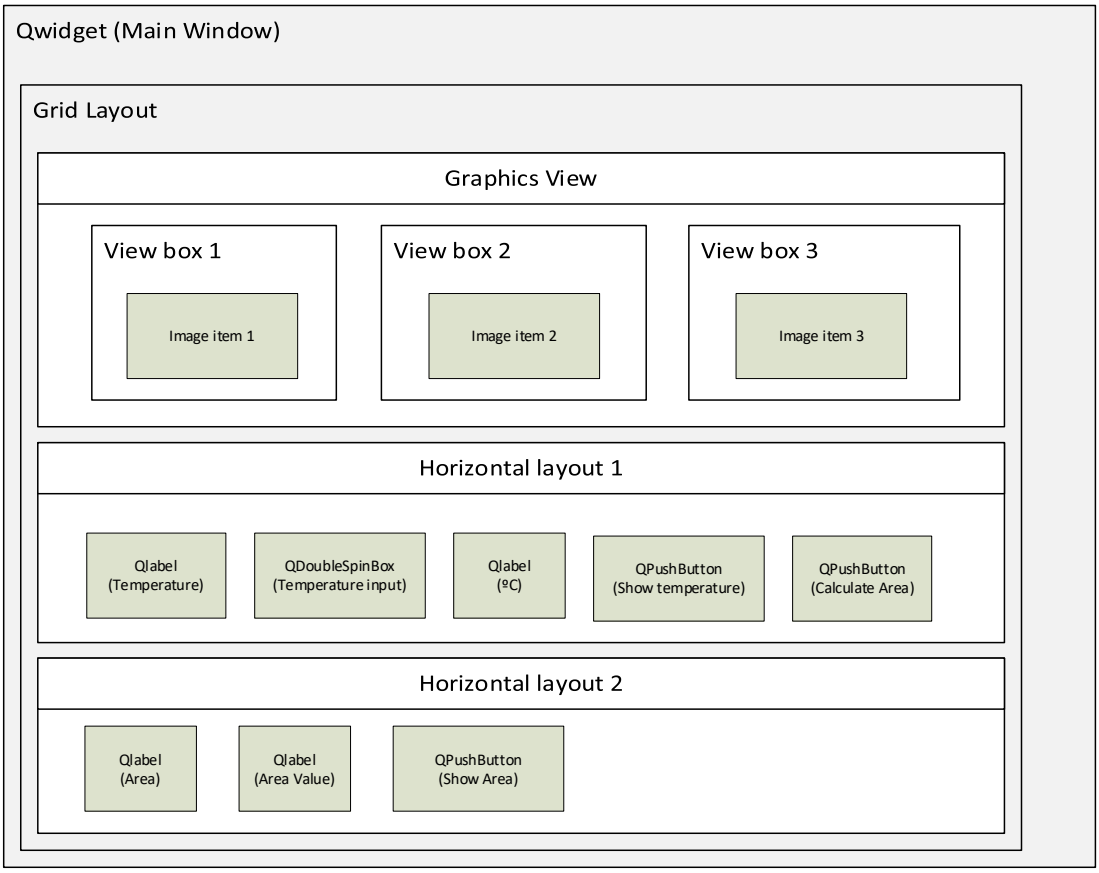


Figura 3.15. Estructura de la GUI.

La GUI cuenta con los siguientes elementos:

- **Ventana principal (Main Window):** Se trata de un widget de Qt para la creación de interfaces gráficas. Permite la inclusión de objetos que permiten la interacción como botones. Los elementos dentro del widget se organizan en layout. Dentro de este elemento se incluye un layout tipo “grid” para organizar los distintos componentes.

- **Graphics view:** Este elemento perime gestionar distintos elementos gráficos 2D. Es necesario para incluir las imágenes que queremos representar. Dentro de este elemento se introduce 1 bloque “**View Box**” para cada una de las imágenes que se incluyen como tipo “**ImageItem**”. Ambos elementos son propios del paquete “pyQtGraph”. Los 3 elementos “ImageItem” tienen las siguientes características:

- **ImageItem 1:** Se trata de la imagen termográfica. Se representa el array con las temperaturas obtenido en la sección 3.2. A esta imagen se le aplica un mapa de colores "Jet" haciendo uso de la función "`setLookupTable`" de "puQtGraph". La definición de las "Look up Tables" de los color maps se encuentran en el Anexo IV.

- **ImageItem 2:** En este elemento se representa el gradiente de temperaturas obtenido en la sección 3.3. De igual forma, se le aplica un mapa de colores, pero en este caso de tipo "Pink".

- **ImageItem 3:** En la última imagen se representa la imagen termográfica en escala de grises superpuesta con las líneas de contorno de 10 temperaturas entre la máxima y la mínima especiadas linealmente.

- **Horizontal layout 1:** Se añade otro tipo de layout para organizar los siguientes elementos:

  - **QLabel (Temperature):** Texto "Temperature:" para indicar donde se introduce la temperatura a la que se quiere calcular el área.

  - **QDoubleSpinBox (Temperature input):** Campo para introducir la temperatura deseada.

  - **QLabel (ºC):** Se muestra el texto "ºC" que indica las unidades de la temperatura.

  - **QPushButton (Show temperature):** Muestra como ayuda la imagen en ese instante con 10 líneas isotermas y su respectiva temperatura en una nueva ventana. También se usa para cerrar dicha ventana.

  - **QPushButton (Calculate Area):** Es un botón al que tiene asignada el cálculo del área bajo la curva de la temperatura indicada en el campo " Temperature input" con el algoritmo descrito en sección 3.5. El algoritmo calcula el área de mayor extensión, aunque también esta implementado para que calcule la suma de las áreas de la misma temperatura.

- **Horizontal layout 2:** En este layout se incluyen 3 elementos:

  - **Qlabel (Area):** Muestra el texto "Area:".

  - **Qlabel (Area Value):** Devuelve el resultado del área calculada.

- **QPushButton (Show Area):** Muestra en una nueva ventana el área de la que se ha calculado el área.

El código para la creación de la GUI se encuentra entre el Anexo III y el Anexo I. El resultado de la interfaz se muestra en la Figura 3.16.

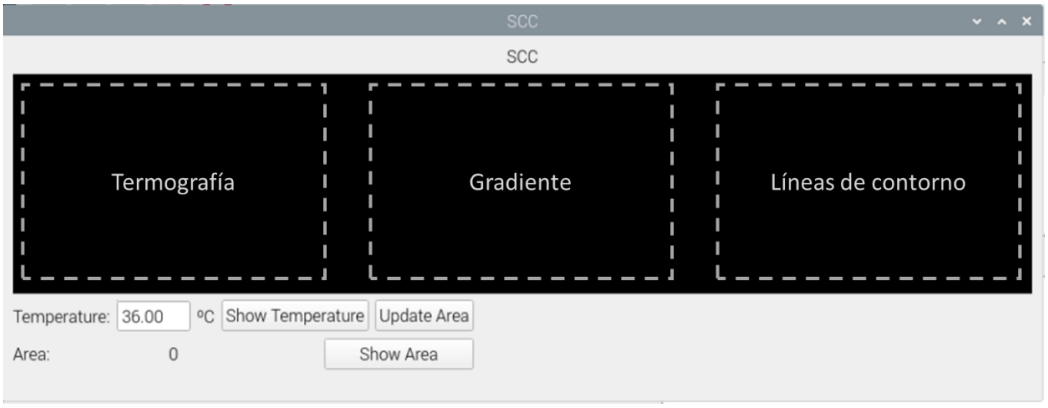


Figura 3.16. Interfaz de la GUI.

3.7 Tiempos de ejecución

En la Tabla 3.1 se muestra un resumen con los tiempos de ejecución medios para cada uno de los algoritmos descritos en esta sección. Estos tiempos se han obtenido tras ejecutar el programa en la Raspberry Pi 3B usada para el desarrollo del trabajo fin de máster.

Tarea	Tiempo
Recibir imagen	18 ms
Convertir imagen a termografía	1 ms
Obtener gradiente	59 ms
Segmentación	11 ms
Cálculo líneas de isotermas	50 ms
Actualización de las imágenes	520ms
Cálculo área	9 ms

Tabla 3.1. Resumen de los tiempos de ejecución de los algoritmos.

El área se calcula cada vez que se solicita y tarda unos 9ms en seleccionar el área con mayor perímetro y calcular su área.

Para la representación de imágenes en tiempo real la Rasperry Pi necesita un tiempo total de unos 659 ms. Con este tiempo de ejecución se consiguen algo menos de 2 frames por segundo. Este es el tiempo más importante puesto que es el necesario para representar las imágenes entiempo real. El tiempo que tardan otras rutinas como la del cálculo del área no es relevante puesto que solo se van a ejecutar en momentos puntuales y nos podemos permitir un tiempo de procesamiento mayor.

4 Test con secuencias de vídeo

Debido a la situación provocada por la COVID-19 no se ha podido probar el funcionamiento del sistema en situaciones reales como era pretendido al inicio del trabajo. En cambio, se han podido hacer test con imágenes reales adquiridas previamente. En este capítulo se muestran ejemplos de pacientes con malformaciones vasculares y hemangiomas, procesando la imagen infrarroja con los algoritmos implementados en el trabajo y haciendo uso de la interfaz desarrollada.

Las imágenes utilizadas han sido adquiridas en consulta siguiendo un protocolo establecido:

- Siempre a la misma distancia y con la misma óptica.
- La temperatura de la sala se mantiene controlada y con poca variación durante el año.
- El sensor se calibra al inicio de cada sesión, tomando una foto de la palma de la mano o la frente y midiendo su temperatura en dos localizaciones distintas.

En los ejemplos del 1 al 5 solo se disponía de una imagen y no de una secuencia de imágenes. Para probar los algoritmos se ha representado en bucle la misma imagen y así poder analizar los datos

En el caso del ejemplo 6 se disponía de una secuencia de 18 imágenes que se han representado una detrás de otra para generar el video.

4.1 Ejemplo 1

En el primer ejemplo se muestra un paciente con una malformación vascular e alto flujo en la zona lumbar. La imagen del espectro visible se muestra en la Figura 4.1. En ella se puede ver una zona en la que la piel toma un color amoratado. Esta sección corresponde a una malformación arterio-venosa y por tanto una malformación de alto flujo sanguíneo. En este tipo de malformaciones se espera un incremento sustancial de la temperatura en la región afectada.

En la Figura 4.2 se muestra el resultado de representar la imagen del espectro infrarrojo con la GUI tras procesarla con los algoritmos implementados. En la zona lumbar se aprecia un área con mayor temperatura que el resto del cuerpo. En la segunda imagen se puede identificar que hay un gradiente de temperatura que delimita la malformación.

Figura 4.1. Imagen del espectro visible el ejemplo 1.

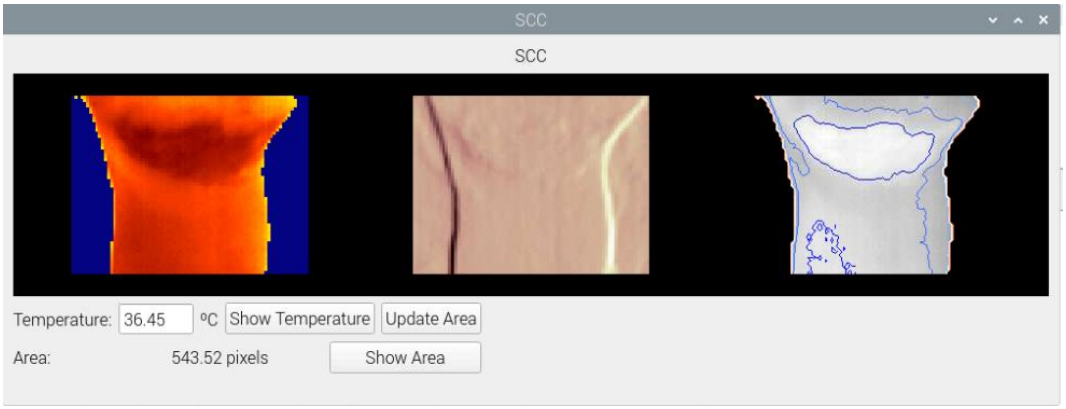
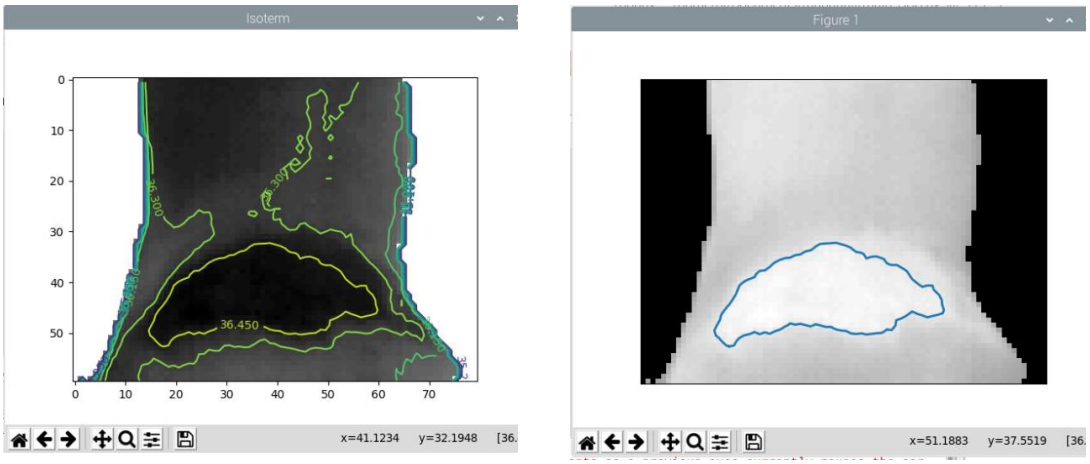


Figura 4.2. Ejemplo 1 representada con la GUI del proyecto.

En la imagen con las curvas de nivel se ve como el área afectada está encerrada en una misma línea isoterma.

Para mostrar las temperaturas del frame actual, pinchamos sobe el botón “Show Temperature” y aparece la ventana que se muestra en la Figura 4.3 (a).



(a) *(b)*

Figura 4.3. (a) imagen con las curvas de nivel con temperaturas del ejemplo 1. (b) área con temperatura de 36.5 ºC del ejemplo 1.

Una vez se ha representado esta imagen, y por tanto se ha adquirido un frame, el botón "Update Area" queda desbloqueado para calcular el área de acuerdo a la temperatura indicada en el campo correspondiente. En este caso el área afectada se encuentra a unos 36.45 ºC. El área calculada a esta temperatura tiene un tamaño de 543 pixeles como se muestra en la Figura 4.2. En la Figura 4.3 (b) se muestra el área delimitada por la línea isoterma de 36.45 ºC. Para que se abra esta ventana es necesario pulsar en el botón "Show Area". El método descrito en la sección 3.5 se trata de un método invariante a rotación lo que permite tomar la imagen desde distintas posiciones y resultado del área es el mismo. Esto es muy útil para el seguimiento de las enfermedades. Se pueden obtener las imágenes en distintos días sin que el área calculada este influida por la imagen tomada.

4.2 Ejemplo 2

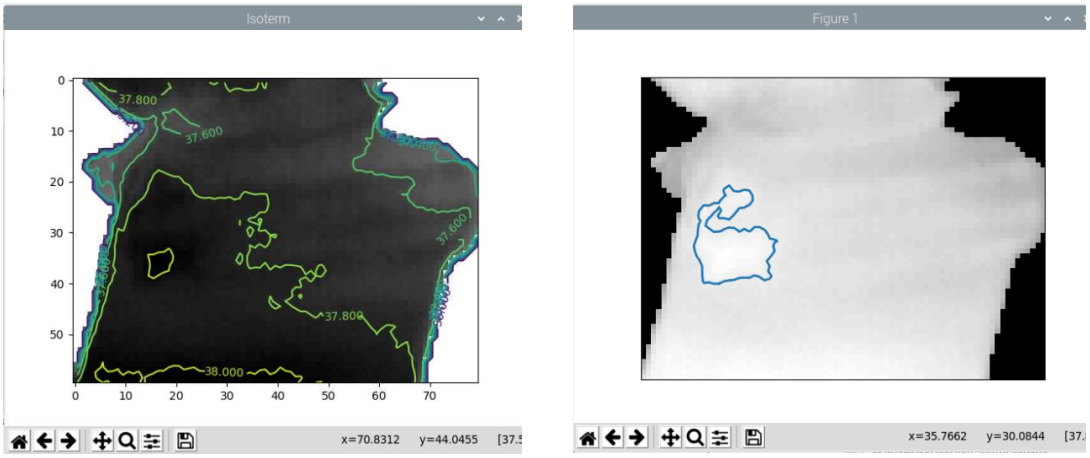
En el siguiente ejemplo se muestra un hemangioma en la espalda. En la Figura 4.4 se muestra la imagen en el espectro visible de la zona afectada.

Figura 4.4. Imagen del espectro visible del ejemplo 2.

De igual manera se representa la imagen con la herramienta creada en el proyecto. Las imágenes infrarrojas procesadas se pueden ver en la Figura 4.5.

Figura 4.5. Ejemplo 2 representada con la GUI del proyecto.

Se aprecia también un incremento de temperatura en la zona afectada que conlleva a un respectivo gradiente de temperatura.



(a) *(b)*

Figura 4.6. (a) imagen con las curvas de nivel con temperaturas del ejemplo 2. (b) área con temperatura de 37.9 ºC del ejemplo 2.

En la Figura 4.6 (a) se aprecia como la malformación se encuentra en una temperatura entre 37.80 y 38 ºC. Se la calcula el área a 37.90 ºC, obteniendo un valor de 200 pixeles. El área correspondiente se muestra en la Figura 4.6 (b).

4.3 Ejemplo 3

Este paciente muestra una mancha de correspondiente a un hemangioma en la cara. La imagen en el espectro visible se muestra en la Figura 4.7.

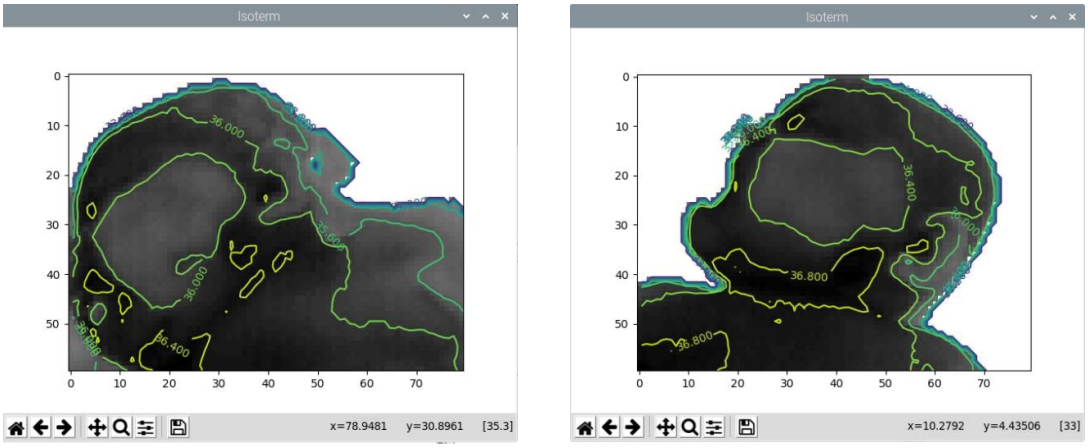
Figura 4.7. Imagen del espectro visible del ejemplo 3.

En este caso el paciente realiza dos visitas en un intervalo de 1 mes. En la Figura 4.8 (a) y (b) se muestran las imágenes del espectro infrarrojo de la primera y segunda visita respectivamente. En la primera visita no se diferencia muy bien la malformación en la imagen. En cambio, en la segunda visita se aprecia que la malformación se encuentra a una temperatura algo superior a 36.80 ºC según la Figura 4.9 (b).

(a)

(b)

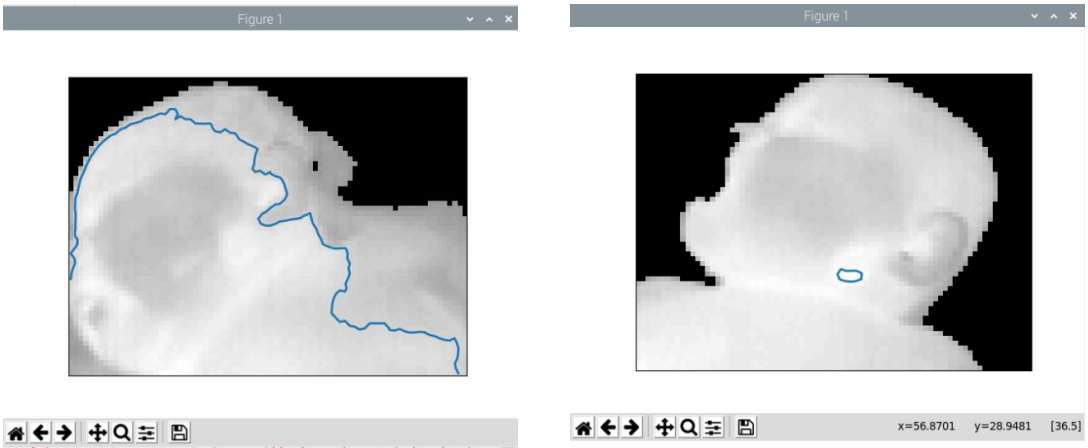
Figura 4.8.. Ejemplo 3 representada con la GUI del proyecto (a) en la primera visita y (b) en la segunda visita.



(a) *(b)*

Figura 4.9. imagen con las curvas de nivel con temperaturas del ejemplo 3 en (a) la primera visita y (b) la segunda visita.

En la segunda visita se estima un área de la malformación de 32 pixeles como se muestra en la Figura 4.8 (b) y la Figura 4.10 (b).



(a) *(b)*

Figura 4.10. Área calculada en (a) la primera visita y (b) la se la segunda visita.

4.4 Ejemplo 4

En este ejemplo se diagnostica un hemangioma en la zona posterior de la cabeza. La imagen adquirida con el sensor RGB se muestra en la

Figura 4.11. Imagen en el espectro visible del ejemplo 4.

Como en los anteriores casos se representa la imagen en el entorno creado en Python. Se muestra en la Figura 4.12. Este es un buen ejemplo para comprobar el algoritmo de segmentación explicado en la sección 3.4.2. Se aprecia como las zonas cubiertas por el cabello son segmentadas al igual que la ropa y el resto de elementos que componen el fondo.

Figura 4.12. Ejemplo 4 representada con la GUI del proyecto.

Se calcula el área a 37.20 ºC siguiendo el mapa de temperaturas de la Figura 4.13 (a). Como resultado se obtiene un área 143 pixeles que se representa en la Figura 4.13 (b).

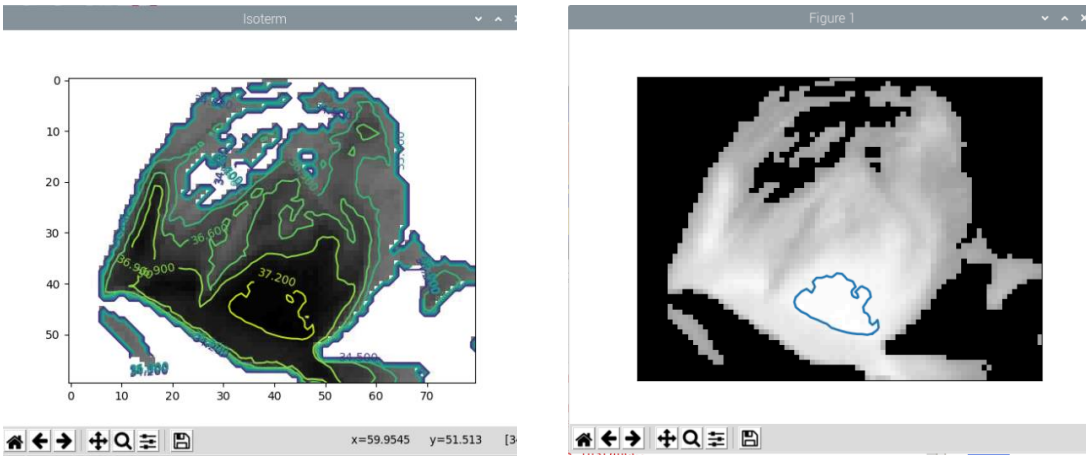


Figura 4.13. (a) imagen con las curvas de nivel con temperaturas del ejemplo 2. (b) área con temperatura de 37.9 ºC del ejemplo 4.

4.5 Ejemplo 5

En este ejemplo se trata la evolución de un paciente afectado por hemangioma en la zona superior de la cabeza. En la Figura 4.14 (a) y (b) se muestran las imágenes en el espectro visible de la primera y segunda visita respectivamente.

En la Figura 4.15 se representan ambas imágenes con los algoritmos y la interfaz del trabajo.

(a) *(b)*

Figura 4.14. Imágenes en el espectro visible de (a) la primera sesión y (b) la segunda sesión del ejemplo 5.

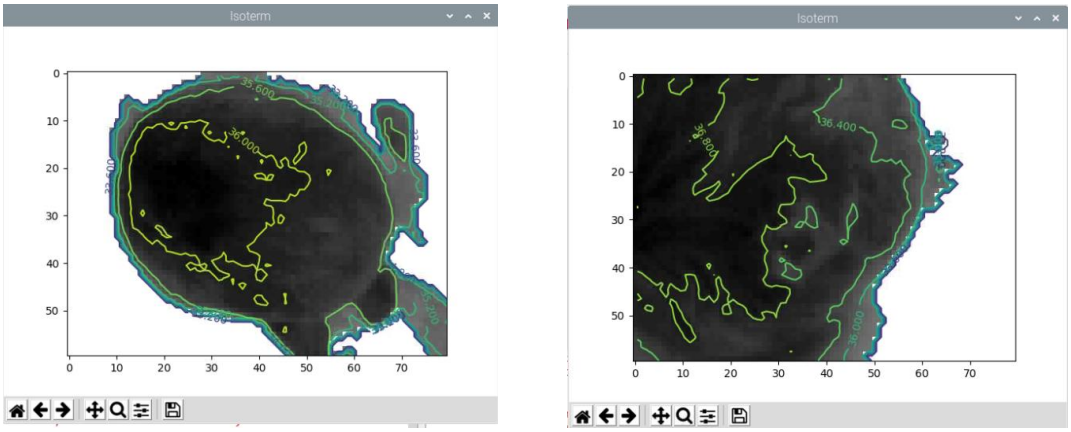
(a)

(b)

Figura 4.15. Ejemplo 5 representada con la GUI del proyecto (a) en la primera visita y (b) en la segunda visita.

Si comparamos las imágenes de ambas visitas se aprecia como el gradiente de temperatura es menor en la segunda visita. Esto indica una mejora en la malformación. De hecho, en las imágenes de la segunda visita no se puede delimitar correctamente el área afectada.

De acuerdo a la Figura 4.16 (a) la temperatura a la que se encuentra la malformación es algo superior a 36 ºC. Se calcula el área a 36.10 ºC obteniendo un resultado de 400 pixeles. En la Figura 4.17 (a) se muestra el área afectada.



(a) *(b)*

Figura 4.16. imagen con las curvas de nivel con temperaturas del ejemplo 5 en (a) la primera visita y (b) la segunda visita.

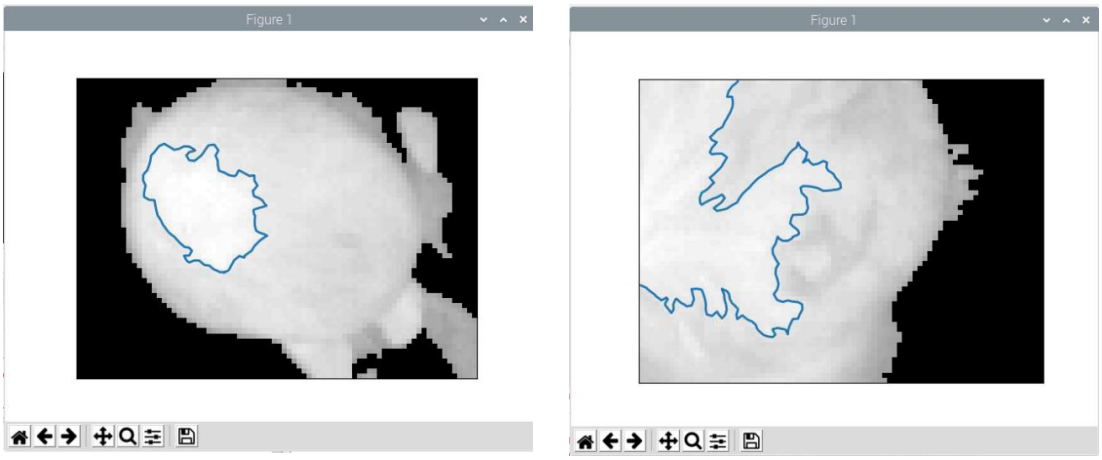


Figura 4.17. Área calculada en (a) la primera visita y (b) la se la segunda visita en el ejemplo 5.

4.6 Ejemplo 6

En este ejemplo se representa una secuencia de video con 18 frames de un hemangioma en la zona superior de la espalda. La imagen en el espectro visible se muestra en la Figura 4.18. En la Figura 4.19 se muestra una captura de la secuencia de video generada con los 18 frames.

Figura 4.18. Imagen en el espectro visible del ejemplo 6.

Figura 4.19. Captura de la secuencia de video de la GIUI con las imágenes del ejemplo 6.

Tras la representación de los 18 frames con el cálculo de 10 líneas de contorno pasa un tiempo promedio de 13.67s, es decir, 0.76s por cada frame. Con este tiempo se consiguen 1.3 fps. Este mismo algoritmo es ejecutado en un ordenador convencional obteniendo un tiempo total de 2.5s, lo que resulta en 7 fps, muy próximo a los 9 fps deseados.

Se realiza otra prueba reduciendo el número e líneas de contorno a calcular de 10 a 5. En este caso se pase un tiempo promedio de 6.93s, es decir, 2.6 fps.

Tras estos 2 test se llega a la conclusión que 2 aspectos a mejorar para obtener una mejor tasa de refresco son: la potencia computacional, cambiando el dispositivo por uno más potente como puede ser la Raspberry 4, y la reducción del número de líneas de contorno a calcular.

Para comprobar que el método de coordenadas es invariante en rotación se calcula el área a con temperatura de 37.60 ºC en 2 frames distintos obteniendo en el primer caso 759 pixeles y en el segundo 756, lo que da un error menor al 0.4%

5 Conclusiones y líneas de futuro

En este capítulo se exponen las conclusiones y los puntos para trabajos futuros tras la realización del trabajo.

En este trabajo se han implementado los algoritmos necesarios para transformar una lista de datos proporcionada por un sensor infrarrojo en una imagen termográfica, así como el cálculo del gradiente y la detección de curvas de nivel con la misma temperatura.

Además de esto se ha implementado una rutina para el cálculo del área de una zona que se encuentra a una temperatura de interés. Tras el análisis de la imagen las imágenes procesadas se puede identificar la temperatura a la que se encuentra la zona afectada por la malformación vascular. Este dato se usa como entrada para aislar la zona deseada y calcular el área correspondiente.

Todo esto se ha integrado en una interfaz gráfica o GUI que hace que estos algoritmos los pueda usar cualquier persona ajena al diseño de la interfaz. En dicha interfaz se muestran al mismo tiempo y en tiempo real la imagen termográfica, el gradiente de temperaturas y las líneas isotermas superpuestas a la imagen infrarroja en escala de grises. En la misma ventana se encuentran los elementos necesarios para identificar la zona afectada y calcular el área.

El cálculo del área de la malformación ayuda al seguimiento de la misma, pudiendo comparar el área y temperatura en distintas sesiones para comprobar su evolución.

Debido a la situación provocada por la COVID-19, estos algoritmos no han podido ser probados en situaciones reales, pero en cambio, se han probado con imágenes reales adquiridas previamente. En dichas pruebas se ha podido comprobar como los algoritmos pueden ayudar en el diagnóstico de las enfermedades.

Como puntos a seguir para futuros trabajos se pueden destacar los siguientes:

- Testeo del sistema adquiriendo imágenes en tiempo real. Es necesario la integración de la interfaz y los algoritmos con el sistema compuesto por la Raspberry Pi y los sensores de imagen, procesando las imágenes en tiempo real.
- Optimización de los algoritmos para alcanzar los 9 fps que es capaz de proporcionar el sensor infrarrojo.
- Implementar los algoritmos en una Raspberry Pi 4, con mayor potencia computacional, y reducir los tiempos de procesado.
- Calcular las líneas isotermas con una separación de tipo logarítmico para así tener mayor número de curvas cerca de la temperatura máxima, a la que se encuentra la malformación, y poder conseguir un cálculo del área más preciso. Además, calculando las líneas de contorno a una temperatura cercana a la máxima se puede reducir el número de líneas de contorno y con ello la carga computacional.
- Implementación de algoritmos para la detección automática del área afectada.
- Utilización de *multithreading* para la reducción de la carga computacional. Un hilo de procesamiento se puede encargar del procesado de las imágenes y otro de representarlas, repartiendo de esta manera el trabajo y siendo más rápido.

6 Bibliografía

[1] A. K. Saxena, G. H. Willita, “Infrared thermography: Experience from a decade of pediatric imaging”

[2] M. T. Garia-Romero, A. Chakkittakandiyil, E. Pope, “The role of infrared thermography in evaluation of proliferative infantile hemangiomas. Results of a pilot study”, Int J Dermatol, 53, 2013.

[3] J. Ayoub Mohammed, A. Balma-Mena, A. Chakkittakandiyil, F. Matea, E. Pope, “Infrared Thermography to Assess Proliferation and Involution of Infantile Hemangiomas. A Prospective Cohort Study”, JAMA Dermatology, 2014.

[4] I. T. Jackson, R. Carreño, Z. Potparic, K. Hussain, “Hemangiomas, vascular malformations, and lymphovenous malformations: classification and methods of treatment”. Plast Reconstr Surg, 1993; 91(07):1216–1230

[5] M. Wassef, F. Blei, D. Adams, A. Alomari, E. Baselga, A. Berenstein et al., “Vascular anomalies classification: Recommendations from the international society for the study of vascular anomalies”, Pediatrics, vol. 136, no. 1, p. 203214, Junio 2015.

[6] J. E. Steiner, B. A. Drolet, “Classification of Vascular Anomalies: An update”, Semin Intervent Radiol, 2017

[7] J. A. Cox, E. Bartlett, E. I. Lee, “Vascular Malformations: A Review”, Semin Plast Surg, 2014.

[8] “ISSVA classification for vascular anomalies”, revisión mayo 2018.

[9] J. A. Leñero-Bardalo, J. Ceballos-Cáceres, Á. Rodriguez-Vazquez, “A custom thermal imaging system for biomedical applications”.

[10] J. A. Leñero-Bardalo, J. Ceballos-Cáceres, Á. Rodriguez-Vazquez, “A customizable thermographic imaging system for medical image acquisition and processing”, Transactions on Biomedical Engineering, 2020

[11] “Raspberry Pi 3: Specs, benchmarks & testing” https://magpi.raspberrypi.org/articles/raspberry-pi-3-specs-benchmarks.

[12] "All Raspberry Pi Products Dimension Drawings" https://www.raspiworld.com/viewtopic.php?t=13.
[13] Manual de la librería "Struct" de Python https://docs.python.org/3/library/struct.html
[14] FLIR LEPTON sensor interface repository https://github.com/groupgets/LeptonModule.
[15] Leptop 2.0 https://groupgets.com/manufacturers/flir/products/lepton-2-0
[16] R. C. Gonzalez and R. E. Woods, *Digital image processing*. 2ª edición, Pearson, 2007.
[17] C. Solomon, T.Breckon. *Fundamentals Of Digital Image Processing: A Practical Approach With Examples In Matlab.* John Wiley & Sons, Inc., 2011.
[18] *Manual de la función "convolve2d" de la librería SciPy https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.convolve2d.html*
[19] I. Arganda-Carreras, C. O. S. Sorzano, P. Thevenaz, A. Muñoz-Barrutia, J. Kybic, R. Marabini, J. M. Carazo, C. Ortiz-de Solorzano, "Non-rigid consistent registration of 2D image sequences", Physics in Medicine and Biology, 2010
[20] J. Yousefi, "Image Binarization using Otsu Thresholding Algorithm", Universidad de Guelph, Ontario, Canada, 2011.
[21] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms", IEEE Trans. systems. Man. and Cybernetics, 1979.
[22] Manual de la libreria "scikit-image" https://scikit-image.org/docs/stable/api/skimage.filters.html#skimage.filters.threshold_otsu
[23] J. C. Webster, *Medical Instrumentation: Application and design*, 4ª edicion, Wiley 2010
[24] Sitio web PyQtGraph http://www.pyqtgraph.org/ , ultimo acceso en 06 de septiembre de 2020.
[25] M. López, Calibración de sensores infrarrojo utilizando la plataforma Raspberry Pi, Trabajo Fin de Grado, Universidad de Sevilla, 2020.
[26] FLIR LEPTON IR cameras manufacturer https://www.flir.com/ , último acceso en 06 de septiembre de 2020.
[27] J.A. Leñero-Bardallo, C. Serrano, B. Acha, J. A. Pérez-Carrasco, J.Bernabeu-Wittel, "Thermography for the differential diagnosis of vascular malformations", Clinical and Experimental Dermatology, 2020.

Anexo I. Main_SCC_video.py

El fichero "Main_SCC_video.py" es el principal del programa. En él se llama a la interfaz gráfica, se actualizan las imágenes y se llama a las funciones que corresponden a los botones de la interfaz. En el inicio del programa se asignan las 3 imágenes a su respectiva posición en la interfaz y las funciones para los botones. Para generar el video se actualizan en un bucle infinito dichas imágenes con una secuencia de 18 frames. En la función `SetIsocurves` Se definen las líneas de contorno que se representan. La función `ShowIsocurve` muestra una ventana con las líneas de contorno, `AskTemperature` solicita el cálculo del área y `ShowArea` muestra el área a la temperatura indicada.

```
import numpy as np
import pyqtgraph as pg
from PyQt5 import QtGui, QtCore
import SCCGUI

import MyColorMaps
import SCC

import time
import numpy as np
import matplotlib.pyplot as plt

path = "/home/pi/Documents/rubpadall/Secuencia de video/"
file = np.array(["raw54.bin", "raw55.bin", "raw56.bin", "raw57.bin",
"raw58.bin", "raw59.bin", "raw60.bin", "raw61.bin", "raw62.bin",
"raw63.bin",
                 "raw64.bin", "raw65.bin", "raw66.bin", "raw67.bin",
"raw68.bin", "raw69.bin",  "raw70.bin", "raw71.bin"])

M=80
N=60
nlevels = 5     # number of isocureves

ColorMap = "flag"
cmap = plt.get_cmap(ColorMap)
```

```
pg.setConfigOptions(imageAxisOrder='row-major')

app = QtGui.QApplication([])

win = QtGui.QMainWindow()
win.setWindowTitle('SCC')
ui = SCCGUI.Ui_MainWindow()
ui.setupUi(win)
win.show()


win2 = pg.GraphicsLayout()
win2.resize(900,350)

vb = win2.addViewBox()
vb2 = win2.addViewBox()
vb3 = win2.addViewBox()

ui.graphicsView.setCentralItem(win2)
ui.graphicsView.resize(900,350)
#TypeError: addItem(self, QGraphicsItem): argument 1 has unexpected
type 'GraphicsLayoutWidget'

vb.setAspectLocked()
imv1 = pg.ImageItem()
imv1.setLookupTable(MyColorMaps.JetLUT, update=True)
vb.addItem(imv1)

vb2.setAspectLocked()
imv2 = pg.ImageItem()
imv2.setLookupTable(MyColorMaps.PinkLUT, update=True)
vb2.addItem(imv2)

vb3.setAspectLocked()
imv3 = pg.ImageItem()
vb3.addItem(imv3)


def SetIsocurve(IM):
    global img, levels, curves, c, init_flag
    #img = pg.ImageItem(IM)


    ## generate empty curves
    curves = []
    levels = np.linspace(IM.min(), IM.max(), nlevels)

    for i in range(len(levels)):
        v = levels[i]
        ## generate isocurve with cmap color selection
        pg_cmap = tuple(int(255*x) for x in cmap(i))
        pg_pen = pg.mkPen(pg_cmap)
        c = pg.IsocurveItem(level=v, pen=pg_pen)
        #c = pg.IsocurveItem(level=v, pen=(i, len(levels)*1.5))
        c.setParentItem(imv3)  ## make sure isocurve is always
correctly displayed over image
        curves.append(c)
    init_flag = 0

        #imgLevels = (IM.min(), IM.max())
```

```
        #imv3.setImage(IM, levels=imgLevels)
        #c.setData(IM)


def UpdateIM():
    global IMSeg


    IM = SCC.IM(A, N, M)
    B = SCC.SF(IM)
    IMSeg = SCC.Otsu(IM)

    imv1.setImage(IMSeg)
    imv2.setImage(B)

    #imv3.clear()
    imgLevels = (IMSeg.min(), IMSeg.max())
    imv3.setImage(IMSeg, levels=imgLevels)
    if (init_flag == 1):
        SetIsocurve(IMSeg)
    for c in curves:
        c.setData(IMSeg)


    #QtGui.QApplication.processEvents()
    app.processEvents()


def ShowIsocurve():
    global IMAnalysis, TempValue, ok
    #TempValue = 0
    if ui.TempBtn.isChecked():
         ui.TempBtn.setText("Continue")
         IMAnalysis = IMSeg.copy()
         ui.AreaBtn.setEnabled(True)
         SCC.PlotIsocurve(IMAnalysis, nlevels, ShowImage='True')

    else:
         ui.TempBtn.setText("Show Temperature")
         SCC.CloseIsocurves(SCC.fig)

def AskTemperature():
    global TempValue

    TempValue = ui.tempSpin.value()

    tarea = time.time()
    Area = SCC.ReqArea(IMAnalysis, TempValue, "greater")

    ui.labelAreaValue.setText('%0.2f pixels' % Area)
    return TempValue

def ShowArea():
    global IMAnalysis, TempValue

    if ui.Showbtn.isChecked():
         ui.Showbtn.setText("Continue")

         IMAnalysis = IMSeg.copy()
```

```
            SCC.PlotRegion(IMAnalysis, ui.tempSpin.value(), ShowImage =
'True', AreaType='greater')

    else:
            ui.Showbtn.setText("Show Area")
            SCC.CloseIsocurves(SCC.figure)



ui.TempBtn.clicked.connect(ShowIsocurve)
ui.AreaBtn.clicked.connect(AskTemperature)
ui.Showbtn.clicked.connect(ShowArea)


i=0
init_flag = 1
ttotal = time.time()
while i<file.shape[0]:
    finput = path + file[i]
    A = SCC.OpenFile(finput)

    UpdateIM()

    i=i+1
    '''
    if (i==file.shape[0]-1) :
        i=0
    '''



print('tiempo total = ', time.time() - ttotal)



## Start Qt event loop unless running in interactive mode or using
pyside.
if __name__ == '__main__':
    import sys
    if (sys.flags.interactive != 1) or not hasattr(QtCore,
'PYQT_VERSION'):
        QtGui.QApplication.instance().exec_()

#Temp = ui.TempSpin.value()
```

Anexo II. SCC.py

En el fichero “SCC.py” se describen todos los algoritmos implementados en el Trabajo Fin de Máster para el procesamiento de imagen. Se puede encontrar más información sobre ellas (argumentos de entrada y lo que devuelven) en los comentarios de cada una de las funciones.

```
import struct
import numpy as np
import math
import scipy
import matplotlib.pyplot as plt

from skimage import data
from skimage import filters
from skimage import exposure
from skimage import measure

K = np.array([  [47, 0, -47],
                [162, 0, -162],
                [47, 0, -47]
             ])  # Sobel-Feldman operator

u = np.array([1, 2, 1])

'''
T2=36.8
T1=36.6

R1=8250
R2=8300
'''

T2=37.1;
T1=36.0;

R1=8250;
R2=8400;
```

```
M=80
N=60

#####################################################################


def OpenFile(finput):
    # Open 'finput' .bin file packaged and return a list 'L'.
Data packaged is unsigned-sorts little-endian
    #reading unsigned shorts 'tuple'??
    with open(finput, "rb") as fid:
        #Max = struct.unpack('H', fid.read(2))          # <little-
endian
        L = list(struct.iter_unpack('<H', fid.read()))
        L = np.asarray(L)                               # tuple to
array
        L = np.reshape(L,L.shape[0],'C')
    return L


def List2Im(L, N=N, M=M, f=3):
    # L: List of data
    # N: number of rows
    # M: number of columns
    # f: first data component of the image.
    #Word 1: Minimum value of the scene
    #Word 2: Maximum value of the scene
    #The rest are 80x60 unsigned shorts, listed by row then
column
    k=f-1

    IM = np.zeros((N,M))

    for i in range(0,N):
        IM[i,:]=L[k+i*M:i*M+M+k]

    return IM


def GetImage(finput, N=N, M=M, f=3):
    # Return a Image as a matrix MxN from finput
    IM = List2Im(OpenFile(finput), N, M, f)
    return IM


def GetImage2(finput):
    A = OpenFile(finput)
    Max = A[0]
    Min = A[1]
    k=3-1

    IM = np.zeros((N,M))

    for i in range(0,N):
        for j in range(0,M):
            IM[i,j]=A[k]
            k=k+1

    '''
    for i in range(0,N):
        IM[i,:]=A[k+i*M:i*M+M+k]
```

```
    '''
    return IM


def IM (A, N=N, M=M, f=3):

    IM = List2Im(A, N, M, f)

    #Now we use the the measured temperature in two body spots to
callibrate the sensor
    a=(T2-T1)/(R2-R1)
    b=T2-a*R2

    IM=IM*a+b


    #MinVal = math.floor(np.amin(IM))
    #MaxVal = math.ceil(np.amax(IM))
    return IM

def IM2 (A, M=M, N=N):

    Max = A[0]
    Min = A[1]
    k=3-1

    IM = np.zeros((N,M))

    for i in range(0,N):
        for j in range(0,M):
            IM[i,j]=A[k]
            k=k+1

    #Now we use the the measured temperature in two body spots to
callibrate the sensor
    a=(T2-T1)/(R2-R1)
    b=T2-a*R2

    IM=IM*a+b

    MinVal = math.floor(np.amin(IM))
    MaxVal = math.ceil(np.amax(IM))
    return IM

def Im2Ter (IM, T1=T1, T2=T2, R1=R1, R2=R2):

    # Transform image IM pixel values to temperature values in °C

    N = IM.shape[0]  # number of rows
    M = IM.shape[1]  # number of columns
    TerIM = np.zeros((N,M))
    #Now we use the the measured temperature in two body spots to
callibrate the sensor
    a=(T2-T1)/(R2-R1)
    b=T2-a*R2

    TerIM=IM*a+b

    #MinVal = math.floor(np.amin(IM))
    #MaxVal = math.ceil(np.amax(IM))
    return TerIM
```

```
def SF(TerIM, Kernel=K):

    MK = K.shape[0]  # number of rows
    NK = K.shape[1]  # number of columns

    M = TerIM.shape[0]  # number of rows
    N = TerIM.shape[1]  # number of columns

    G = np.zeros((M,N))

    #G = scipy.signal.convolve2d(TerIM, Kernel, mode='same',
boundary='fill', fillvalue=0)
    G = scipy.signal.convolve2d(TerIM, np.flip(Kernel),
boundary='symm', mode='same')
    '''
    for i in range(MK-2,M-MK+1):
        for j in range (NK-2,N-NK+1):
            #B[i,j]=np.sum(np.sum(K*A(range(i-((MK-1)//2),i+((MK-
1)//2)),range(j-((NK-1)//2),j+((NK-1)//2)))))
            G[i,j]=np.sum(K*TerIM[i-((MK-1)//2):i+((MK-
1)//2)+1,j-((NK-1)//2):j+((NK-1)//2)+1])

    '''
    # MinTemp = np.min(A)
    # MaxTemp = np.max(A)

    # MaxVar=MaxTemp-MinTemp

    MinVal = math.floor(np.min(G))
    MaxVal = math.ceil(np.max(G))

    M = G.shape[0]  # number of rows
    N = G.shape[1]  # number of columns
    #t = time.time()

    for i in range(0, M):
        for j in range(0, N):

            if G[i,j]>0:
                G[i,j]=G[i,j]/MaxVal#*(MaxVar/MaxVal)
            else:
                G[i,j]=-G[i,j]/MinVal#*(MaxVar/MinVal)

    return G

def sobelkernel(size, varargin=[]):
    #SOBELKERNEL returns sobel kernel
    #   KERNEL=SOBELKERNEL(SIZE) Returns Sobel filter of
predefined size.
    #
    #   KERNEL=SOBELKERNEL(SIZE, 'NORMALISED') The Sobel matrix
should be
    #   normalised if proper derivative estimator is required.
    #
    #   [KERNEL, S, D]=SOBELKERNEL(SIZE, 'NORMALISED') Returns
also smoothing S
    #   and derivative D components individually. The kernel is
then a
    #   multiplication of these two vectors: S'*D. This can be
usefull for
```

```
    #   convolution acceleration via separable kernels as
illustrated at:
    #
http://blogs.mathworks.com/steve/2006/10/04/separable-
convolution/
    #
    #   Example
    #   -------
    #       sobelkernel(4)
    #
    #   See also IMFILTER, FSPECIAL.
    #   Contributed by Jan Motl (jan@motl.us)
    #   $Revision: 1.1 $  $Date: 2013/02/13 16:58:01 $
    #   For method description see:
    #       http://stackoverflow.com/questions/9567882/sobel-
filter-kernel-of-large-size
    # Parameter checking.


    # Parameter checking.
    if (len(varargin)!=0) and (varargin == 'normalise'):
        normalisation = 1/8
    else:
        normalisation = 1

    # The dafault 3x3 Sobel kernel.
    s = normalisation * u;
    d = np.array([1, 0, -1])

    # Convolve the default 3x3 kernel to the desired size.
    for i in range(0, size-3):
        s = normalisation * np.convolve(u, s,)
        d = np.convolve(u, d)
    #kernel = np.dot(x,d);

    kernel = np.zeros((s.shape[0],d.shape[0]))

    for i in range(0, s.shape[0]):
        for j in range(0, d.shape[0]):
            kernel[i,j]=s[i]*d[j]

    return kernel

def Otsu (IM, thr="lower"):

    val = filters.threshold_otsu(IM)
    #hist, bins_center = exposure.histogram(IM)
    N = IM.shape[0]  # number of rows
    M = IM.shape[1]  # number of columns

    #IMOtsu = IM
    IMOtsu = IM.copy()
    MinVal = math.floor(np.amin(IM))
    MaxVal = math.ceil(np.amax(IM))

    if (thr == "lower"):

        for i in range(0,N):
            for j in range(0,M):
                if (IMOtsu[i,j]<val):
                    IMOtsu[i,j]=MinVal
```

```
    elif(thr == "upper"):
        for i in range(0,N):
            for j in range(0,M):
                if (IMOtsu[i,j]>val):
                    IMOtsu[i,j]=MinVal


    return IMOtsu


def CoordinateMethods(xi, yi):
    #xi: array with x coordinates of a isocurve
    #yi: array with y coordinates of a isocurve
    S1 = xi[1]*yi[1+1]
    S2 = xi[1+1]*yi[1]
    for i in range (0, xi.shape[0]-1):
        S1 = S1 + xi[i]*yi[i+1]
        S2 = S2 + xi[i+1]*yi[i]

    Area = abs(S1-S2)/2

    return Area

def SiftingArea(Arrays):
    size_Arrays = np.shape(Arrays)
    #print('shape = ', size_Arrays[0])
    n_Arrays = np.zeros(size_Arrays[0])

    for i in range(size_Arrays[0]):
        Array_x = Arrays[i]
        n_Arrays [i] = Array_x.shape[0]

    max_index = np.where(n_Arrays==(np.max(n_Arrays)))
    max_index = (np.asarray(max_index[0]))
    np.reshape(max_index,max_index.shape[0],'C')
    return int(max_index)

def ReqArea(IM, level, AreaType='sum'):
    Area = 0
    if (level < np.amin(IM) or level > np.amax(IM)):
        return 0
    else:
        ContourCordinates = measure.find_contours(IM, level)
#Retunrs (row, column) array
        #↕ContourCordinates =
ContourCordinates[(SiftingArea(ContourCordinates))]
        size_contour = np.shape(ContourCordinates)
        CC = np.zeros(size_contour[0])

        if (AreaType == "sum"):
            for i in range(size_contour[0]):
                A = ContourCordinates[i]

                Area = Area + CoordinateMethods(A[:,1], A[:,0])
# xi = column, yi =row
        elif (AreaType == "greater"):
            SiftingArea(ContourCordinates)
            A = ContourCordinates[SiftingArea(ContourCordinates)]
            Area = CoordinateMethods(A[:,1], A[:,0])
```

```
        return Area

def  PlotIsocurve(IM, nlevels, ShowImage = 'False'):
    global fig

    fig = plt.figure("Isoterm")
    ax = fig.add_subplot(111)


    if ShowImage == 'True':
        ax.imshow(IM, cmap='Greys')

    cont = plt.subplot()
    cs = cont.contour(IM, nlevels, origin='lower')
    ax.clabel(cs)

    plt.show()

def CloseIsocurves(Fig):

    plt.close(Fig)

def PlotRegion(IM, level, ShowImage = 'False', AreaType='sum'):
    global figure
    contours = measure.find_contours(IM, level)
    # Display the image and plot all contours found
    figure, ax = plt.subplots()
    if ShowImage == 'True':
        ax.imshow(IM, cmap=plt.cm.gray)


    if (AreaType=="sum"):
        for n, contour in enumerate(contours):
            ax.plot(contour[:, 1], contour[:, 0], linewidth=2)
    elif (AreaType=="greater"):
        contours = contours[(SiftingArea(contours))]
        ax.plot(contours[:, 1], contours[:, 0], linewidth=2)

    ax.axis('image')
    ax.set_xticks([])
    ax.set_yticks([])
    plt.show()
```

Anexo III. SCCGUI.py

En el archive "SCCGUI.py" se implementa una clase crea la interfaz descrita en la sección 3.6.

```
#SCCGUI
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1000, 350)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.gridLayout =
QtWidgets.QGridLayout(self.centralwidget)
        self.gridLayout.setObjectName("gridLayout")

        #self.page = QtWidgets.QWidget()
        #self.page.setObjectName("page")

        #self.gridLayout_2 =
QtWidgets.QGridLayout(self.centralwidget)
        #self.gridLayout_2.setObjectName("gridLayout_2")
        self.graphicsView = GraphicsView(self.centralwidget)
        self.graphicsView.setObjectName("graphicsView")
        self.gridLayout.addWidget(self.graphicsView, 1, 0, 1, 0)
        #self.gridLayout_2.resize(950,25)

        spacerItem = QtWidgets.QSpacerItem(40, 20,
QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)
        self.gridLayout.addItem(spacerItem, 4, 3, 1, 1)

        self.titleLabel = QtWidgets.QLabel(self.centralwidget)
        font = QtGui.QFont()
        font.setPointSize(12)
        self.titleLabel.setFont(font)
        self.titleLabel.setAlignment(QtCore.Qt.AlignCenter)
        self.titleLabel.setObjectName("titleLabel")
```

```
        self.gridLayout.addWidget(self.titleLabel, 0, 0, 1, 0)

        self.horizontalLayout = QtWidgets.QHBoxLayout()
        self.horizontalLayout.setObjectName("horizontalLayout")

        self.labelTemp = QtWidgets.QLabel(self.centralwidget)
        self.labelTemp.setObjectName("labelTemp")
        self.horizontalLayout.addWidget(self.labelTemp)

        self.tempSpin =
QtWidgets.QDoubleSpinBox(self.centralwidget)

self.tempSpin.setButtonSymbols(QtWidgets.QAbstractSpinBox.NoButto
ns)
        self.tempSpin.setProperty("value", 36)
        self.tempSpin.setObjectName("tempSpin")
        self.horizontalLayout.addWidget(self.tempSpin)

        self.labelC = QtWidgets.QLabel(self.centralwidget)
        self.labelC.setObjectName("label")
        self.horizontalLayout.addWidget(self.labelC)

        self.TempBtn = QtGui.QPushButton('Show Temperature')
        self.TempBtn.setObjectName("TempBtn")
        self.TempBtn.setCheckable(True)
        self.horizontalLayout.addWidget(self.TempBtn)

        self.AreaBtn = QtGui.QPushButton('Update Area')
        self.AreaBtn.setObjectName("AreaBtn")
        self.AreaBtn.setCheckable(False)
        self.AreaBtn.setEnabled(False)
        self.horizontalLayout.addWidget(self.AreaBtn)

        self.horizontalLayout_2 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_2.setObjectName("horizontalLayout")

        self.gridLayout.addLayout(self.horizontalLayout, 2, 0, 1,
1)

        self.labelArea = QtWidgets.QLabel(self.centralwidget)
        self.labelArea.setObjectName("labelArea")
        self.horizontalLayout_2.addWidget(self.labelArea)
        self.labelAreaValue =
QtWidgets.QLabel(self.centralwidget)
        self.labelAreaValue.setObjectName("labelAreaValue")
        self.horizontalLayout_2.addWidget(self.labelAreaValue)

        self.Showbtn = QtGui.QPushButton('Show Area')
        self.Showbtn.setObjectName("Showbtn")
        self.Showbtn.setCheckable(True)
        self.horizontalLayout_2.addWidget(self.Showbtn)

        self.gridLayout.addLayout(self.horizontalLayout_2, 3, 0,
1, 1)


        MainWindow.setCentralWidget(self.centralwidget)


        self.retranslateUi(MainWindow)
```

```
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow",
"SCC"))
        self.titleLabel.setText(_translate("MainWindow", "SCC"))
        self.labelTemp.setText(_translate("MainWindow",
"Temperature:"))
        self.labelC.setText(_translate("MainWindow", "ยบC"))
        self.labelArea.setText(_translate("MainWindow", "Area:"))
        self.labelAreaValue.setText(_translate("MainWindow",
"0"))

from pyqtgraph import GradientWidget, GraphicsView, SpinBox
#from pyqtgraph.widgets.RawImageWidget import RawImageWidget
```

Anexo IV. MyColorMaps.py

Los mapas de colores utilizados se definen como Look Up Tables con las herramientas de lalibreria “pyQtGraph” en el archivo “MyColorMaps.py”. Para ello se cogen 6 puntos cos los colores RGB de dichos mapas de colores.

```
import numpy as np
import pyqtgraph as pg
from PyQt5 import QtGui, QtCore

#Setting Jet an Pink colormaps
Jetcolors = [
    (5, 5, 130),
    (0, 96, 255),
    (32, 255, 223),
    (255, 255, 0),
    (255, 64, 0),
    (128, 0, 0)
    ]
cmapJet = pg.ColorMap(pos=np.linspace(0.0, 1.0, 6),
color=Jetcolors)
JetLUT = cmapJet.getLookupTable(alpha=False)

Pinkcolors = [
    (30, 0, 0),
    (141, 91, 91),
    (197, 138, 131),
    (219, 199, 162),
    (237, 237, 196),
    (255, 255, 255)
    ]
cmapPink = pg.ColorMap(pos=np.linspace(0.0, 1.0, 6),
color=Pinkcolors)
PinkLUT = cmapPink.getLookupTable(alpha=False)
```

Anexo V. Manual de uso

En este anexo se comenta como se utilizan los programas desarrollados. En la ruta "/home/pi/Documents/rubpadall/SCC" (Figura V.1) se encuentran los archivos necesarios para correr el programa. Dentro de esta carpeta también se incluyen las imágenes utilizadas para generar la secuencia de video del Anexo IV.

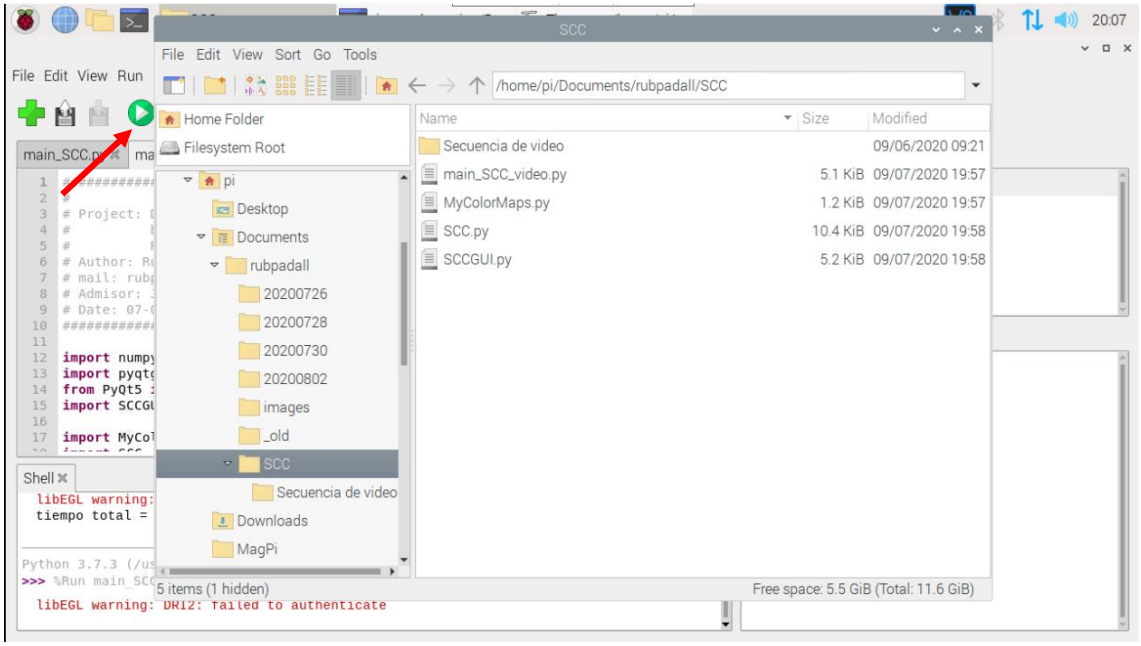


Figura V.1. Ruta del programa.

Para correr el programa se abre el archivo "main_SCC_video.py" con el IDE ThonnyPy y se pulsa sobre el botón "Ejecutar" marcado con una flecha roja en la Figura V.1. Es importante que los archivos estén en la misma estructura.

Los archivos también se pueden encontrar en el repositorio de GitHub https://github.com/RPadial/SCC.

La utilización del entorno es muy sencilla. Nada más ejecutar el programa se muestra la interfaz y se genera el video con las 3 imágenes. Para mostrar las líneas isotermas con la temperatura es necesario pulsar en el botón “Show Temperature” como se muestra en la Figura V.2. Tras pulsar sobre el botón el video se congela y texto del botón cambia a “Continue”. Para continuar con el video es necesario pulsar sobre este botón, que cierra la ventana con las curvas de nivel. La ventana que se abre corresponde a la que se muestra en la Figura V.3 (a)

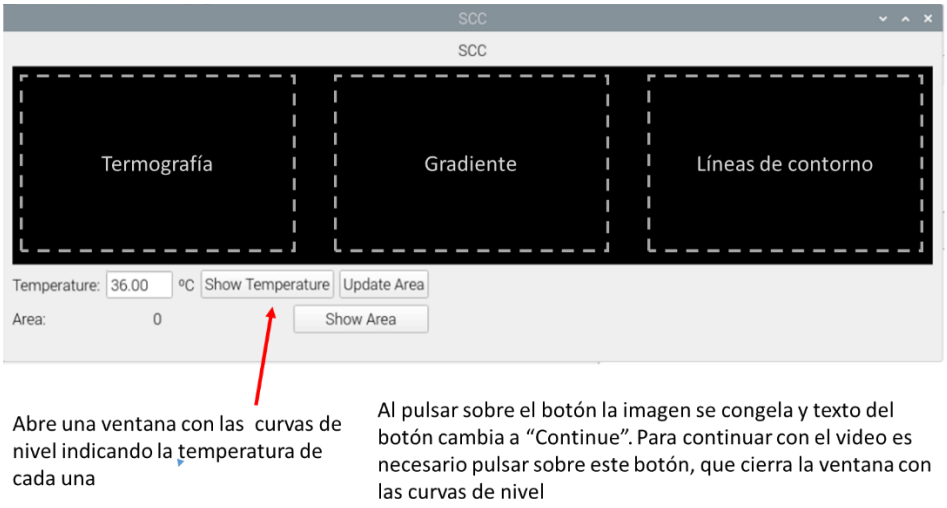
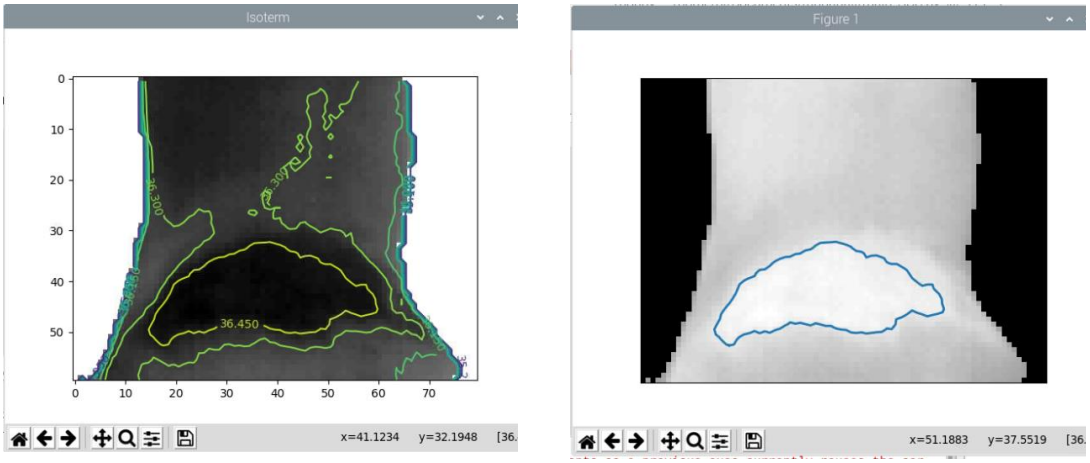


Figura V.2. Primer paso para la utilización de la interfaz.



(a) *(b)*

Figura V.3. (a) imagen con las curvas de nivel con temperaturas del ejemplo 1. (b) área con temperatura de 36.5 ºC del ejemplo 1.

Una vez se identifica la temperatura de la que se quiere calcular el área se introduce en el campo correspondiente. Tras esto se pulsa sobre el botón “Update Area” que calcula el área en base a la temperatura seleccionada y la imagen adquirida al pulsar sobre el botón “Show Temperature”. Para mostrar el área de la temperatura de interés se pulsa sobre el botón “Show Area” y se muestra una ventana como la de la Figura V.3 (b). Para cerrar la ventana, al igual que antes, se debe pulsar sobre el botón que ahora tiene el texto “Continue”. El resultado del área se representa en el campo específico. Los pasos se resumen en la Figura V.4 y

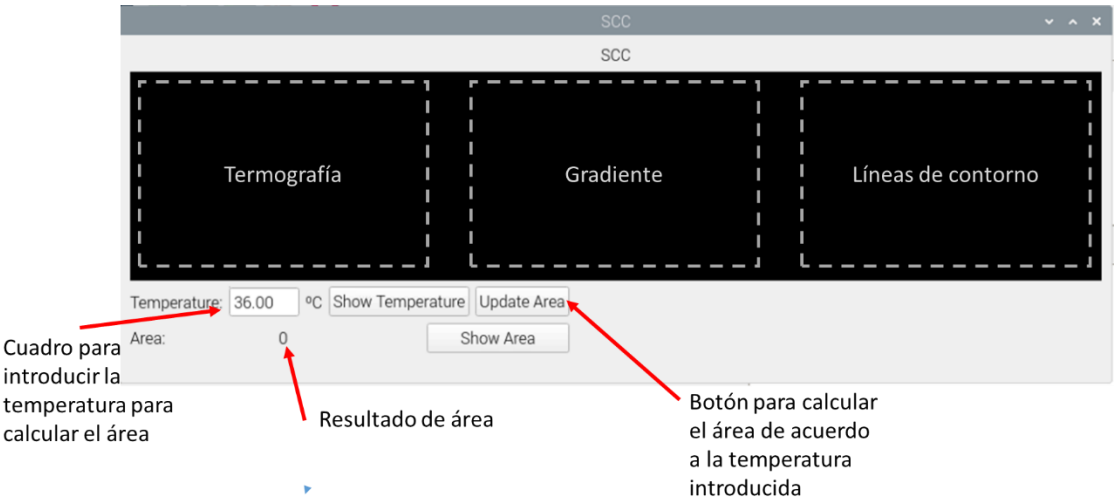


Figura V.4. Paso 2 para el uso de la interfaz.

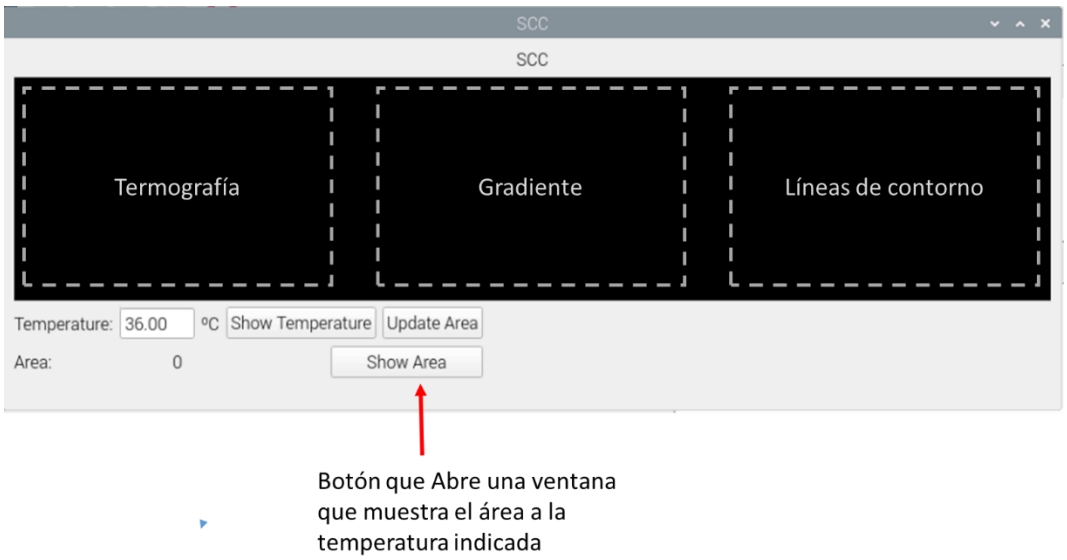


Figura V.5. Paso 3 para el uso de la interfaz.