



UNIVERSIDAD
DE ALMERÍA



TRABAJO FINAL TRATAMIENTO DIGITAL DE IMÁGENES

TÍTULO:

ANÁLISIS AUTOMATIZADO DE IMGÁGENES DE TAC
CRANEAL PARA LA DETECCIÓN Y CARACTERIZACIÓN
DE HEMATOMAS EN CONTEXTO CLÍNICO

AUTOR:

LUCAS BARRIENTOS MUÑOZ

ALMERÍA
24 DE MAYO DE 2024

ÍNDICE

1	INTRODUCCIÓN	1
2	ANTECEDENTES.....	2
2.1	IMPORTANCIA DEL TRAUMATISMO CRANEOENCEFÁLICO (TCE).....	2
2.2	TIPOS DE HEMATOMAS INTRACRANEALES	2
2.2.1	HEMATOMA EPIDURAL (HE).....	2
2.2.2	HEMATOMA SUBDURAL (HS)	2
2.2.3	HEMORRAGIA SUBARACNOIDEA (HSA).....	2
2.3	EPIDEMIOLOGÍA DEL TCE EN LA POBLACIÓN MAYOR DE 65 AÑOS	3
2.4	HALLAZGOS TEMPRANOS Y TARDÍOS EN TC DE PACIENTES CON TCE LEVE A MODERADO	3
2.5	INCIDENCIA DE HEMORRAGIAS INTRACRANEALES EN PACIENTES CON TCE LEVE	3
2.6	RELEVANCIA Y JUSTIFICACIÓN DEL ESTUDIO	4
3	FUNDAMENTOS TEÓRICOS.....	4
3.1	INTRODUCCIÓN A LA TOMOGRAFÍA COMPUTARIZADA (TC).....	4
3.2	FUNCIONAMIENTO Y COMPONENTES DEL TAC	5
3.3	ESCALA DE HOUNSFIELD	5
3.3.1	FUNDAMENTOS DE LA ESCALA DE HOUNSFIELD	5
3.3.2	RANGOS DE DENSIDAD EN LA ESCALA DE HOUNSFIELD.....	5
3.3.3	APLICACIÓN EN IMÁGENES DE TC.....	6
3.3.4	IMPORTANCIA CLÍNICA.....	6
3.3.5	EJEMPLOS DE USO.....	6
3.3.6	VENTAJAS Y LIMITACIONES	7
3.3.6.1	VENTAJAS.....	7
3.3.6.2	LIMITACIONES.....	7
4	METODOLOGÍA.....	7
4.1	ESTRUCTURA DEL CÓDIGO	7
4.1.1	FUNCIÓN RESALTAR 246.....	8
4.1.2	FUNCIÓN CREAR CARPETA SALIDA	8
4.1.3	FUNCIÓN ELIMINAR PÍXELES BAJA DENSIDAD	9
4.1.4	FUNCIÓN APLICAR FILTRO MEDIANA	10
4.1.5	FUNCIÓN GUARDAR MATRIZ HIGHLIGHTED	11
4.1.6	FUNCIÓN GUARDAR IMAGEN HIGHLIGHTED	11
4.1.7	FUNCIÓN CREAR MÁSCARA BINARIA	12
4.1.8	FUNCIÓN ENCONTRAR CONTORNOS	13
4.1.8.1	INICIALIZACIÓN Y PREPARACIÓN	13
4.1.8.2	FUNCIONES LAMBDA PARA VALIDACIÓN Y DETECCIÓN DE BORDES	13
4.1.8.3	BUCLE PRINCIPAL PARA ENCONTRAR CONTORNOS.....	14
4.1.9	FUNCIÓN CLASIFICAR HEMATOMA	14
4.1.10	FUNCIÓN PROCESAR IMAGEN	16
4.1.11	FUNCIÓN MAIN	17
5	ESTUDIO DE LA IMPLEMENTACIÓN	18
6	CONCLUSIONES	18
7	BIBLIOGRAFÍA	18

ÍNDICE DE FIGURAS

Figura 1: Diagrama de flujo.....	7
Figura 2: Función Resaltar246.....	8
Figura 3: Función CrearCarpetaSalida.....	9
Figura 4: Función EliminarPíxelesBajaDensidad.....	9
Figura 5: Función AplicarFiltroMediana.....	10
Figura 6: Función GuardarMatrizHighlighted.....	11
Figura 7: Función GuardarImagenHighlighted.....	12
Figura 8: Función CrearMáscaraBinaria.....	12
Figura 9: Inicialización EncontrarContornos.....	13
Figura 10: Funciones Lambda EncontrarContornos.....	13
Figura 11: Bucle para encontrar contornos de la función EncontrarContornos.....	14
Figura 12: Función ClasificarHematoma.....	15
Figura 13: Función ProcesarImagen.....	17
Figura 14: Función Main.....	17

ÍNDICE DE TABLAS

....

LISTADO DE ACRÓNIMOS

TAC Tomografía Axial Computarizada.

TCE Traumatismo Craneoencefálico.

HU Unidades Hounsfield.

1 INTRODUCCIÓN

En las últimas décadas, el avance de las tecnologías médicas ha transformado radicalmente el diagnóstico y tratamiento de diversas condiciones neurológicas. Una de las herramientas más importantes en este campo es la **Tomografía Axial Computarizada (TAC)** craneal, una técnica de imagen no invasiva que permite obtener cortes transversales detallados del cerebro. Esta herramienta es esencial para la detección de patologías intracraneales, tales como los hematomas, que son acumulaciones de sangre fuera de los vasos sanguíneos y que pueden representar emergencias médicas graves.

Los hematomas intracraneales se clasifican principalmente en tres tipos: epidurales, subdurales e intracerebrales. Cada uno de estos tipos presenta características específicas en las imágenes de **TAC**, y su identificación precisa y oportuna es crucial para la gestión adecuada del paciente. La correcta caracterización de estos hematomas puede significar la diferencia entre una intervención quirúrgica exitosa y un resultado desfavorable para el paciente.

El diagnóstico y caracterización de hematomas a partir de imágenes de **TAC** ha dependido tradicionalmente del ojo experto de radiólogos y neurólogos. Sin embargo, la interpretación humana está sujeta a errores, y la creciente demanda de servicios médicos requiere soluciones que puedan mejorar la precisión y eficiencia de los diagnósticos. En este contexto, la inteligencia artificial y el aprendizaje automático han emergido como tecnologías prometedoras para automatizar y mejorar la interpretación de imágenes médicas.

Este trabajo se centra en el desarrollo de un programa en **C++** que analiza imágenes de **TAC** craneal para la detección y caracterización automática de hematomas. El objetivo principal es crear una herramienta que asista a los profesionales de la salud en la identificación precisa y rápida de los diferentes tipos de hematomas, proporcionando así una segunda opinión automatizada que pueda mejorar la toma de decisiones clínicas.

El desarrollo de este programa involucra varios desafíos técnicos, incluyendo la segmentación de imágenes, la extracción de características relevantes y la clasificación de los tipos de hematomas. Se utilizan técnicas avanzadas de procesamiento de imágenes y algoritmos de aprendizaje automático para abordar estos desafíos. La implementación se valida mediante un conjunto de imágenes de **TAC** con anotaciones realizadas por expertos, evaluando la precisión y fiabilidad del sistema en diferentes escenarios clínicos.

Este documento se estructura de la siguiente manera: en el apartado de **Antecedentes** se revisan los estudios previos y tecnologías existentes en el campo de la detección automatizada de hematomas. El apartado de **Fundamentos Teóricos** proporciona una descripción detallada de los TAC craneales y los diferentes tipos de hematomas, estableciendo el contexto clínico y técnico necesario para entender el problema y las soluciones propuestas. La **Metodología** describe el desarrollo del programa, incluyendo los algoritmos utilizados y los procesos de preprocesamiento y análisis de imágenes. En el **Estudio de la Implementación** se presentan los resultados obtenidos, las pruebas realizadas y una discusión sobre la eficacia del sistema. Finalmente, en las **Conclusiones** se resumen los hallazgos del trabajo, se evalúa la efectividad del programa y se proponen futuras líneas de investigación.

En conclusión, este trabajo tiene como objetivo contribuir al campo de la radiología y la neurología mediante el desarrollo de una herramienta automatizada que no solo aumente la precisión en la detección de hematomas intracraneales, sino que también alivie la carga de trabajo de los profesionales médicos, mejorando así la calidad de la atención al paciente.

2 ANTECEDENTES

2.1 IMPORTANCIA DEL TRAUMATISMO CRANEOENCEFÁLICO (TCE)

El **traumatismo craneoencefálico (TCE)** es una de las principales causas de mortalidad y morbilidad a nivel mundial, afectando a millones de personas cada año. Este tipo de lesión es especialmente prevalente en la población anciana debido a factores como la mayor propensión a caídas y la fragilidad biológica asociada con el envejecimiento. La creciente población de personas mayores de 65 años implica que el **TCE** seguirá siendo un desafío significativo para los sistemas de salud globales.

2.2 TIPOS DE HEMATOMAS INTRACRANEALES

Existen varios tipos de hematomas intracraneales que pueden resultar de un **TCE**, entre los cuales se destacan los **hematomas epidurales**, **subdurales** y la **hemorragia subaracnoidea**. Cada uno de estos tipos presenta características y desafíos específicos en términos de diagnóstico y tratamiento.

2.2.1 HEMATOMA EPIDURAL (HE)

Características: El HE ocurre cuando se acumula sangre entre el cráneo y la duramadre, generalmente debido a una ruptura de la arteria meníngea media. Es más común en jóvenes y está frecuentemente asociado con fracturas craneales.

Diagnóstico: La detección temprana mediante tomografía computarizada (TC) es crucial, ya que el HE puede expandirse rápidamente y causar un aumento en la presión intracraneal.

Tratamiento: Requiere intervención quirúrgica inmediata para evacuar el hematoma y aliviar la presión.

2.2.2 HEMATOMA SUBDURAL (HS)

Características: El HS se forma cuando la sangre se acumula entre la duramadre y la aracnoides, generalmente como resultado de la ruptura de venas corticales. Es más común en ancianos y personas que toman anticoagulantes.

Diagnóstico: La TC es la herramienta principal para la detección del HS, que puede presentarse como agudo, subagudo o crónico.

Tratamiento: Dependiendo de la gravedad y el tamaño del hematoma, el tratamiento puede variar desde la observación y manejo conservador hasta la evacuación quirúrgica.

2.2.3 HEMORRAGIA SUBARACNOIDEA (HSA)

Características: La HSA se produce cuando hay sangrado en el espacio subaracnoideo, a menudo debido a la ruptura de un aneurisma cerebral o a un trauma severo. Puede causar irritación meníngea y un aumento en la presión intracraneal.

Diagnóstico: La TC es fundamental para detectar la HSA y evaluar la extensión del sangrado.

Tratamiento: Puede incluir manejo médico para controlar la presión intracraneal y, en casos de aneurismas, intervención quirúrgica o endovascular.

2.3 EPIDEMIOLOGÍA DEL TCE EN LA POBLACIÓN MAYOR DE 65 AÑOS

El estudio de la epidemiología del **TCE** en la población mayor de 65 años a lo largo de 25 años en un hospital universitario español de tercer nivel proporciona una visión detallada de los cambios en los mecanismos de lesión y las tendencias en las comorbilidades asociadas. Los hallazgos clave incluyen un aumento significativo en las caídas como la principal causa de **TCE**, pasando de representar el **8.33%** en el periodo **1991-1995** a más del **70%** en **2011-2015**, y una disminución en los accidentes de tráfico como causa de TCE.

Además, el estudio observó un incremento en la proporción de pacientes con hematoma epidural, de un **1.39%** en **1990-1995** a un **9.46%** en **2010-2015**. Estos cambios reflejan la necesidad de adaptar las estrategias de prevención y tratamiento del **TCE** para esta población vulnerable, enfatizando la importancia de una vigilancia continua y de técnicas avanzadas de diagnóstico.¹

2.4 HALLAZGOS TEMPRANOS Y TARDÍOS EN TC DE PACIENTES CON TCE LEVE A MODERADO

La evaluación de los cambios entre las **tomografías computarizadas (TC)** iniciales y tardías en pacientes con **TCE** leve a moderado es crucial para entender la progresión de la lesión y la necesidad de intervenciones médicas. Un estudio realizado en Turquía, que incluyó **2644 pacientes**, encontró que el **4.24%** de estos pacientes presentaron cambios significativos entre las **TC iniciales** y **tardías**. De estos, el **30%** mostró **deterioro neurológico** y el **41.1%** requirió **cirugía** basada en los hallazgos de las **TC tardías**. Este estudio subraya la importancia de las **TC** seriadas para modificar las decisiones de tratamiento en un subgrupo de pacientes.²

2.5 INCIDENCIA DE HEMORRAGIAS INTRACRANEALES EN PACIENTES CON TCE LEVE

Un estudio retrospectivo realizado en el University Hospital Bern, Suiza, investigó la incidencia de hemorragias intracraneales (HIC) en pacientes con TCE leve, definido por una puntuación de la Escala de Coma de Glasgow (GCS) de 14-15. De los 3088 pacientes incluidos, el 4.8% presentó HIC en las TC iniciales. A pesar de que ninguno de estos pacientes mostró deterioro neurológico significativo o requirió intervención neuroquirúrgica urgente, la incidencia de HIC aumentó con la edad y el uso de anticoagulantes.

¹ Chicote Álvarez, E., González Castro, A., Ortiz Lasa, M., Jiménez Alfonso, A., Escudero Acha, P., Rodríguez Borregán, J. C., Peñasco Martín, Y., & Dierssen Sotos, T. (2018). Epidemiología del traumatismo craneoencefálico en la población mayor de 65 años a lo largo de 25 años. *Revista española de anestesiología y reanimación*, 65(10), 546–551. <https://doi.org/10.1016/j.redar.2018.06.003>

² Dalbayrak, S., Gumustas, S., Bal, A., & Akansel, G. (2011). *Early and delayed CT findings in patients with mild-to-moderate head trauma*. *Turkish neurosurgery*, 21(4), 591–598.

Este estudio es fundamental para entender los riesgos asociados con el TCE leve, especialmente en poblaciones vulnerables como los ancianos. La identificación temprana de HIC puede influir significativamente en las decisiones clínicas, mejorando los resultados y reduciendo las complicaciones a largo plazo.³

2.6 RELEVANCIA Y JUSTIFICACIÓN DEL ESTUDIO

La detección y caracterización automatizada de hematomas epidurales, subdurales y la hemorragia subaracnoidea en imágenes de TAC craneal es de vital importancia por varias razones:

1. **Precisión Diagnóstica:** Los sistemas automatizados pueden aumentar la precisión y rapidez del diagnóstico, reduciendo el margen de error humano y proporcionando una herramienta de apoyo valiosa para los radiólogos y neurólogos.
2. **Reducción de la Carga de Trabajo:** En un entorno clínico con alta demanda, las herramientas automatizadas pueden aliviar la carga de trabajo de los profesionales de la salud, permitiéndoles enfocarse en casos más complejos y en la atención directa al paciente.
3. **Mejora en los Resultados Clínicos:** La detección temprana y precisa de hematomas puede mejorar significativamente los resultados clínicos, facilitando intervenciones oportunas y adecuadas que pueden salvar vidas y reducir las secuelas a largo plazo.
4. **Optimización de Recursos:** La implementación de sistemas de detección automatizada puede optimizar el uso de recursos hospitalarios, minimizando la necesidad de estudios adicionales y prolongadas observaciones clínicas.

3 FUNDAMENTOS TEÓRICOS

3.1 INTRODUCCIÓN A LA TOMOGRAFÍA COMPUTARIZADA (TC)

La tomografía computarizada (TC) es una técnica de imagen médica esencial que utiliza rayos X y procesamiento computarizado para crear imágenes detalladas del interior del cuerpo humano. Desde su desarrollo en 1972 por Godfrey Hounsfield, la TC ha revolucionado el diagnóstico médico, proporcionando imágenes tridimensionales que permiten una evaluación precisa de diversas patologías. Este método supera las limitaciones de la radiografía convencional al proporcionar imágenes de alta resolución y detalladas, lo que es crucial para el diagnóstico de enfermedades intracraneales, cardiovasculares, oncológicas y muchas otras condiciones.

La evolución de la TC ha incluido avances significativos como la TC helicoidal y la TC multidetector, que permiten la adquisición continua y rápida de imágenes de alta resolución. Estos avances han mejorado notablemente la capacidad de diagnóstico y han reducido el tiempo necesario para obtener imágenes completas del cuerpo, lo que es crucial en situaciones de emergencia, como el traumatismo craneoencefálico (TEMA 5: TOMOGRAFÍA COMPUTERIZADA).

³ Albers, C. E., von Allmen, M., Evangelopoulos, D. S., Zisakis, A. K., Zimmermann, H., & Exadaktylos, A. K. (2013). *What is the incidence of intracranial bleeding in patients with mild traumatic brain injury? A retrospective study in 3088 Canadian CT head rule patients.* BioMed research international, 2013, 453978. <https://doi.org/10.1155/2013/453978>

3.2 FUNCIONAMIENTO Y COMPONENTES DEL TAC

Un escáner de TC consiste en un gantry (una estructura circular) que contiene un tubo de rayos X giratorio y detectores de radiación. Durante el escaneo, el tubo de rayos X gira alrededor del paciente, emitiendo haces de radiación que atraviesan el cuerpo. Los detectores de radiación, situados en el lado opuesto del gantry, absorben y miden la radiación restante después de que haya atravesado los tejidos. Un programa informático reconstruye estas mediciones en imágenes detalladas del interior del cuerpo.

El proceso de adquisición de imágenes en TC implica la toma de múltiples cortes o "rebanadas" del cuerpo, que luego se ensamblan para formar una imagen tridimensional. Esto permite una visualización detallada de estructuras internas complejas y una evaluación precisa de cualquier anomalía presente. La capacidad de ajustar los parámetros del escáner, como la energía de los rayos X y el tiempo de exposición, permite optimizar la calidad de las imágenes y minimizar la dosis de radiación al paciente (TEMA 5: TOMOGRAFÍA COMPUTERIZADA).

3.3 ESCALA DE HOUNSFIELD

La escala de Hounsfield, también conocida como unidades Hounsfield (UH), es una medida cuantitativa utilizada en tomografía computarizada (TC) para describir la densidad radiológica de los tejidos en comparación con el agua y el aire. Esta escala es fundamental para interpretar las imágenes de TC, ya que permite diferenciar entre distintos tipos de tejidos en base a su densidad relativa.

3.3.1 FUNDAMENTOS DE LA ESCALA DE HOUNSFIELD

La escala de Hounsfield se basa en el coeficiente de atenuación de los tejidos, que es la capacidad de un material para absorber o atenuar los rayos X. Los valores en la escala de Hounsfield están definidos de la siguiente manera:

Agua: 0 UH, utilizado como referencia estándar.

Aire: -1000 UH, debido a su baja densidad y capacidad de atenuación.

Hueso compacto: +1000 UH o más, debido a su alta densidad y capacidad de atenuación.

3.3.2 RANGOS DE DENSIDAD EN LA ESCALA DE HOUNSFIELD

Cada tipo de tejido tiene un rango característico en la escala de Hounsfield, lo que facilita su identificación en las imágenes de TC. A continuación se presentan los rangos de densidad de algunos tejidos y materiales comunes:

Aire: -1000 UH.

Grasa: -100 a -50 UH.

Tejido pulmonar: -700 a -600 UH.

Tejido adiposo subcutáneo: -120 a -80 UH.

Líquido cerebroespinal: -15 a +15 UH.

Tejido blando (músculo, cerebro): +20 a +80 UH.

Tejido renal: +20 a +40 UH.

Sangre: +30 a +45 UH.

Hígado: +40 a +60 UH.

Sangre hiperdensa (blanca): +60 a +100 UH.

Hueso esponjoso: +300 a +400 UH.

Hueso cortical: +700 a +3000 UH.

3.3.3 APLICACIÓN EN IMÁGENES DE TC

En una imagen de **TC**, estos valores de densidad se representan en una escala de grises. Los diferentes niveles de gris corresponden a los valores de **UH** de los **tejidos**, lo que permite a los radiólogos y médicos identificar y diferenciar las estructuras anatómicas y las posibles patologías.

Negro: Representa el aire, con valores cercanos a -1000 UH.

Gris oscuro: Representa la grasa, con valores negativos menores que 0 UH.

Gris claro: Representa los tejidos blandos como músculos y órganos internos, con valores positivos menores que 100 UH.

Blanco: Representa el hueso y otras estructuras densas, con valores superiores a 1000 UH.

3.3.4 IMPORTANCIA CLÍNICA

La precisión en la diferenciación de tejidos utilizando la escala de Hounsfield es crucial en la práctica clínica. Por ejemplo, en el contexto del **traumatismo craneoencefálico (TCE)**, la **escala de Hounsfield** permite distinguir entre los diferentes tipos de **hematomas** (epidural, subdural, subaracnoideo) y otras **lesiones intracraneales**.

1. **Hematomas Epidurales:** Aparecen como áreas hiperdensas (blancas) con valores altos en la escala de Hounsfield debido a la alta densidad de la sangre acumulada.
2. **Hematomas Subdurales:** También son hiperdensos en la fase aguda, pero pueden volverse isodensos o hipodensos en fases subagudas y crónicas.
3. **Hemorragia Subaracnoidea:** Se presenta como áreas hiperdensas en los surcos cerebrales y las cisternas basales, indicando la presencia de sangre en el espacio subaracnoideo.

3.3.5 EJEMPLOS DE USO

Diagnóstico de Tumores: Los tumores suelen tener densidades específicas que los diferencian de los tejidos circundantes. Por ejemplo, las metástasis hepáticas pueden aparecer como áreas hipodensas en comparación con el tejido hepático normal.

Evaluación de Fracturas Óseas: Los huesos fracturados aparecen como discontinuidades en las áreas hiperdensas del hueso cortical.

Detección de Enfermedades Pulmonares: La TC de tórax puede revelar áreas de consolidación (neumonía) o nódulos pulmonares, que se diferencian del tejido pulmonar normal por sus valores de Hounsfield.

3.3.6 VENTAJAS Y LIMITACIONES

3.3.6.1 VENTAJAS

Precisión Diagnóstica: La escala de Hounsfield proporciona una medida cuantitativa que mejora la precisión del diagnóstico.

Versatilidad: Se puede aplicar a una amplia variedad de patologías y tejidos.

No Invasiva: Permite una evaluación interna detallada sin necesidad de procedimientos invasivos.

3.3.6.2 LIMITACIONES

Variabilidad entre Pacientes: Los valores pueden variar ligeramente entre pacientes debido a diferencias individuales en la composición de los tejidos.

Artefactos de Imagen: La presencia de artefactos, como los causados por movimiento o materiales metálicos, puede afectar la precisión de los valores de Hounsfield.

4 METODOLOGÍA

En este apartado, se detalla la metodología empleada para el desarrollo del programa de análisis automatizado de imágenes de tomografía axial computarizada (TAC) craneal para la detección y caracterización de hematomas. A continuación, se presenta una explicación detallada de cada uno de los métodos implementados en el código proporcionado, estructurados paso a paso y explicando la lógica detrás de cada uno de ellos.

4.1 ESTRUCTURA DEL CÓDIGO

El programa está compuesto por varias funciones clave que permiten procesar las imágenes de TAC, resaltar ciertas características, eliminar ruido, encontrar contornos, y finalmente clasificar el tipo de hematoma presente. El flujo del programa se puede resumir en los siguientes pasos principales:

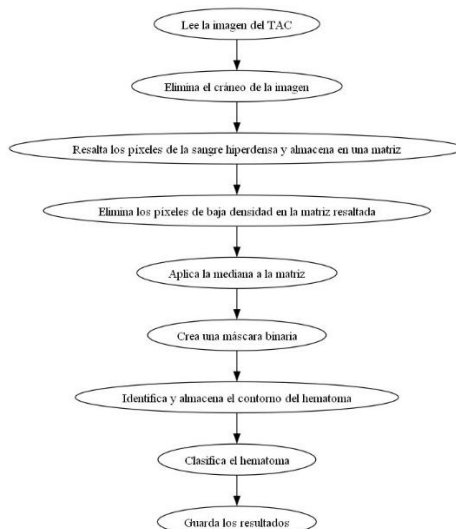
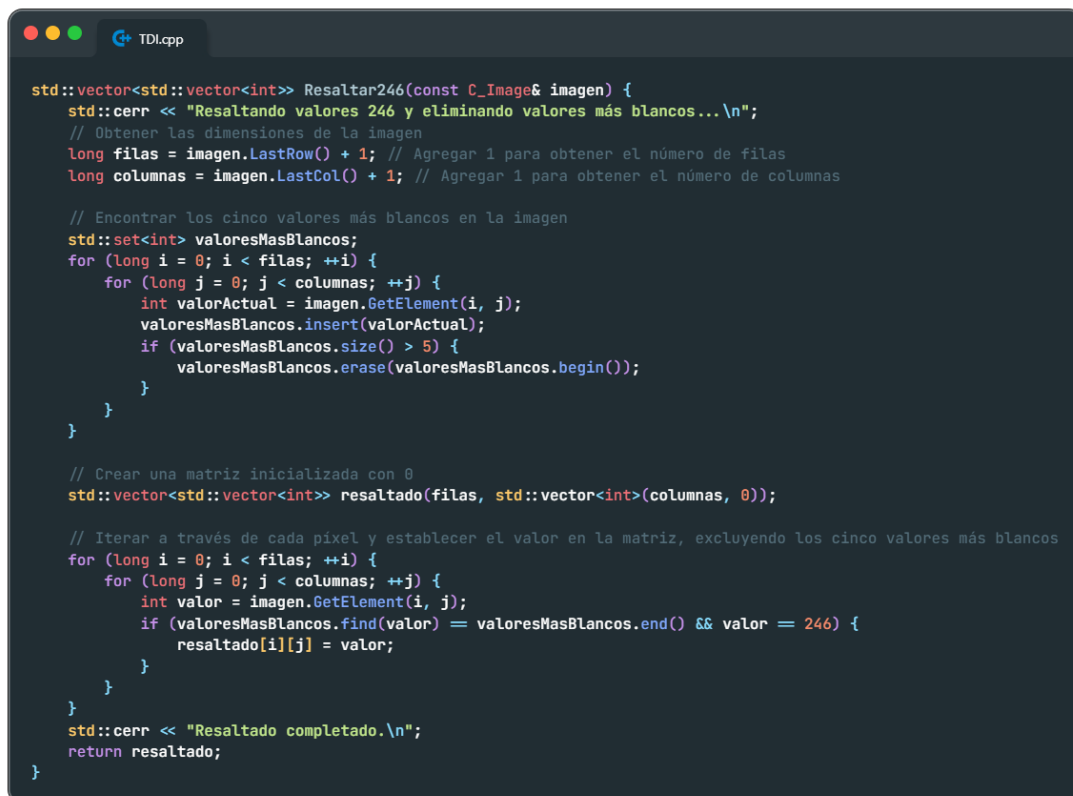


Figura 1: Diagrama de flujo.

4.1.1 FUNCIÓN RESALTAR 246

Esta función se encarga de resaltar los píxeles con valores 246, este color es el color predominante en los hematomas usados para las pruebas, dado que no se ha utilizado una base muy grande para realizarlas, a la hora de llevar este algoritmo a producción, habría que realizar un estudio para adaptarlo a cualquier tipo de imagen, determinando un rango de tonos de gris que consiga una mayor eficacia.

En esta función, además, se ha implementado un apartado que detecta cuáles son los 5 tonos más blancos en la imagen, es decir, los más altos del rango 0 – 255, para posteriormente poner su valor en 0, así se consigue eliminar el cráneo en las imágenes de tal forma que no se resalte ninguno de sus valores y altere el resultado de la imagen.



```
std::vector<std::vector<int>> Resaltar246(const C_Image& imagen) {
    std::cerr << "Resaltando valores 246 y eliminando valores más blancos...\n";
    // Obtener las dimensiones de la imagen
    long filas = imagen.LastRow() + 1; // Agregar 1 para obtener el número de filas
    long columnas = imagen.LastCol() + 1; // Agregar 1 para obtener el número de columnas

    // Encontrar los cinco valores más blancos en la imagen
    std::set<int> valoresMasBlancos;
    for (long i = 0; i < filas; ++i) {
        for (long j = 0; j < columnas; ++j) {
            int valorActual = imagen.GetElement(i, j);
            valoresMasBlancos.insert(valorActual);
            if (valoresMasBlancos.size() > 5) {
                valoresMasBlancos.erase(valoresMasBlancos.begin());
            }
        }
    }

    // Crear una matriz inicializada con 0
    std::vector<std::vector<int>> resultado(filas, std::vector<int>(columnas, 0));

    // Iterar a través de cada píxel y establecer el valor en la matriz, excluyendo los cinco valores más blancos
    for (long i = 0; i < filas; ++i) {
        for (long j = 0; j < columnas; ++j) {
            int valor = imagen.GetElement(i, j);
            if (valoresMasBlancos.find(valor) == valoresMasBlancos.end() && valor == 246) {
                resultado[i][j] = valor;
            }
        }
    }

    std::cerr << "Resaltado completado.\n";
    return resultado;
}
```

Figura 2: Función Resaltar246.

Como se puede observar en la figura, se ha implementado un método **GetElement()** no existente en la librería, de tal forma que se devuelva el valor de la matriz directamente desde la imagen.

4.1.2 FUNCIÓN CREAR CARPETA SALIDA

Esta función crea una carpeta al instante de ejecutar el código, lo que hará que todos los archivos generados por este se guarden en esa carpeta. Se comprueba el repositorio contenedor, que en este caso será la carpeta **"Run"**, y se verifica que no exista ninguna otra carpeta con ese nombre, que será **"PruebaN"**, donde **N** será el número de prueba, en caso de que exista alguna carpeta con este mismo nombre, se creará una nueva donde **N** será **N+1**.



```

std::string CrearCarpetaSalida() {
    std::cerr << "Creando carpeta de salida...\n";
    std::string baseNombre = "Prueba";
    int numero = 1;
    std::string nombreCarpeta;

    while (true) {
        nombreCarpeta = baseNombre + std::to_string(numero);
        if (!fs::exists(nombreCarpeta)) {
            fs::create_directory(nombreCarpeta);
            break;
        }
        ++numero;
    }
    std::cerr << "Carpeta creada: " << nombreCarpeta << "\n";
    return nombreCarpeta;
}

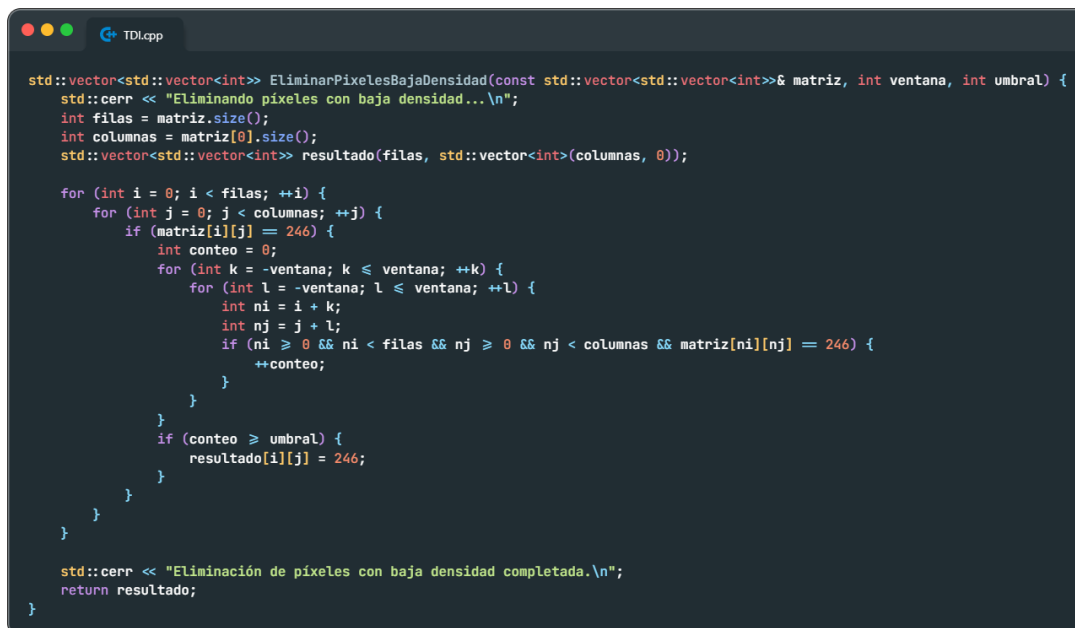
```

Figura 3: Función *CrearCarpetaSalida*.

En esta función, se utiliza la biblioteca estándar de C++ para la manipulación del sistema de archivos a través del espacio de nombres **std::filesystem**, abreviado como **fs**. Esta biblioteca proporciona una interfaz para trabajar con el sistema de archivos, lo que incluye operaciones como la comprobación de la existencia de directorios y archivos, así como la creación de directorios.

4.1.3 FUNCIÓN ELIMINAR PÍXELES BAJA DENSIDAD

Esta función se utiliza para eliminar los píxeles de baja densidad de una matriz de imagen, específicamente aquellos que no forman parte de un hematoma concentrado. Esta función es esencial para limpiar la imagen de TAC craneal, eliminando los bordes del cráneo y reduciendo el ruido que no es relevante para la detección del hematoma. La función toma como parámetros una matriz de enteros representando la imagen, un tamaño de ventana y un umbral de densidad.



```

std::vector<std::vector<int>> EliminarPíxelesBajaDensidad(const std::vector<std::vector<int>>& matriz, int ventana, int umbral) {
    std::cerr << "Eliminando píxeles con baja densidad...\n";
    int filas = matriz.size();
    int columnas = matriz[0].size();
    std::vector<std::vector<int>> resultado(filas, std::vector<int>(columnas, 0));

    for (int i = 0; i < filas; ++i) {
        for (int j = 0; j < columnas; ++j) {
            if (matriz[i][j] == 246) {
                int conteo = 0;
                for (int k = -ventana; k ≤ ventana; ++k) {
                    for (int l = -ventana; l ≤ ventana; ++l) {
                        int ni = i + k;
                        int nj = j + l;
                        if (ni ≥ 0 && ni < filas && nj ≥ 0 && nj < columnas && matriz[ni][nj] == 246) {
                            ++conteo;
                        }
                    }
                }
                if (conteo ≥ umbral) {
                    resultado[i][j] = 246;
                }
            }
        }
    }

    std::cerr << "Eliminación de píxeles con baja densidad completada.\n";
    return resultado;
}

```

Figura 4: Función *EliminarPíxelesBajaDensidad*.

El proceso comienza con la creación de una matriz de salida, inicialmente llena de ceros. Luego, para cada píxel con un valor de **246** (indicativo de sangre hiperdensa), se cuenta cuántos píxeles dentro de una ventana definida alrededor de ese píxel también tienen el valor **246**. Si el número de píxeles con valor **246** dentro de esta ventana es mayor o igual al umbral especificado, el píxel se conserva en la matriz de salida. De lo contrario, se elimina (se establece a cero). Este método permite diferenciar áreas densamente pobladas de píxeles relevantes, como los hematomas, de áreas con ruido disperso o píxeles que forman parte del borde del cráneo, que generalmente están menos concentrados. De esta manera, se mejora la precisión de la detección y caracterización del hematoma en la imagen procesada.

4.1.4 FUNCIÓN APLICAR FILTRO MEDIANA

Esta función se emplea para aplicar un filtro de mediana a una matriz de imagen, lo que ayuda a reducir el ruido y a resaltar mejor las áreas afectadas, como los hematomas en las imágenes de **TAC** craneal. Este proceso es crucial para mejorar la calidad de la imagen y facilitar la detección precisa de los hematomas. La función toma como entrada una matriz de enteros que representa la imagen y utiliza la clase `C_Matrix` de la biblioteca proporcionada para realizar el filtrado.



```
std::vector<std::vector<int>>> AplicarFiltroMediana(const std::vector<std::vector<int>>& matriz) {
    std::cerr << "Aplicando filtro de mediana...\n";
    C_Matrix matrizOriginal(0, matriz.size() - 1, 0, matriz[0].size() - 1);
    C_Matrix matrizFiltrada(0, matriz.size() - 1, 0, matriz[0].size() - 1);

    for (size_t i = 0; i < matriz.size(); ++i) {
        for (size_t j = 0; j < matriz[0].size(); ++j) {
            matrizOriginal(i, j) = matriz[i][j];
        }
    }

    matrizFiltrada.MedianFilter(matrizOriginal, 1);

    std::vector<std::vector<int>>> resultadoFiltrado(matriz.size(), std::vector<int>(matriz[0].size()));
    for (size_t i = 0; i < matriz.size(); ++i) {
        for (size_t j = 0; j < matriz[0].size(); ++j) {
            resultadoFiltrado[i][j] = matrizFiltrada(i, j);
        }
    }

    std::cerr << "Filtro de mediana aplicado.\n";
    return resultadoFiltrado;
}
```

Figura 5: Función `AplicarFiltroMediana`.

El proceso comienza creando dos objetos `C_Matrix`: `matrizOriginal` y `matrizFiltrada`, que se inicializan con las dimensiones de la matriz de entrada. Luego, se copian los valores de la matriz de entrada en `matrizOriginal`. La función `MedianFilter`, que es parte de la biblioteca incluida, se aplica a `matrizOriginal` y el resultado se almacena en `matrizFiltrada`. El filtro de mediana es efectivo para eliminar el ruido porque reemplaza cada píxel con la mediana de los valores en su vecindario, preservando los bordes mientras suaviza las variaciones menores.

Finalmente, los valores filtrados se copian de `matrizFiltrada` a una nueva matriz de `std::vector<std::vector<int>>>`, que se devuelve como resultado. Este método asegura que la imagen procesada esté libre de ruido y que las áreas de interés, como

los hematomas, estén claramente definidas, facilitando así su detección y análisis en pasos posteriores del procesamiento de imágenes.

4.1.5 FUNCIÓN GUARDAR MATRIZ HIGHLIGHTED

Esta función se encarga de guardar la matriz resaltada en un archivo de texto. Esta matriz, que contiene los píxeles de interés de una imagen de **TAC** craneal, es esencial para el análisis posterior y la detección de hematomas. Al almacenar esta matriz en un archivo, se facilita la revisión, el procesamiento adicional y el archivo de los datos procesados.

El proceso de la función comienza abriendo un flujo de salida de archivo (**std::ofstream**) con la ruta especificada por el parámetro **rutaArchivoSalida**. Si el archivo no puede ser abierto, se imprime un mensaje de error y la función termina prematuramente. Esto asegura que no se intente escribir en un archivo no accesible.

Una vez abierto el archivo, se determinan las dimensiones de la matriz highlighted (**número de filas y columnas**). La función luego itera sobre cada elemento de la matriz, escribiendo los valores en el archivo de salida. Los valores en cada fila se separan por tabulaciones (**\t**) y cada fila se termina con una nueva línea (**\n**). Este formato estructurado permite que la matriz sea fácilmente leída y analizada por otras herramientas o programas.



```

void GuardarMatrizHighlighted(const std::vector<std::vector<int>>& highlighted, const std::string& rutaArchivoSalida) {
    std::cerr << "Guardando matriz highlighted en archivo...\n";
    std::ofstream outFile(rutaArchivoSalida);
    if (!outFile) {
        std::cerr << "Error al abrir el archivo de salida: " << rutaArchivoSalida << std::endl;
        return;
    }

    long filas = highlighted.size();
    long columnas = highlighted[0].size();

    for (long i = 0; i < filas; ++i) {
        for (long j = 0; j < columnas; ++j) {
            outFile << highlighted[i][j];
            if (j < columnas - 1) {
                outFile << "\t";
            }
        }
        outFile << std::endl;
    }

    outFile.close();
    std::cerr << "Matriz guardada en archivo.\n";
}

```

Figura 6: Función GuardarMatrizHighlighted.

Finalmente, **se cierra el flujo de archivo** y se imprime un mensaje indicando que la matriz ha sido guardada exitosamente. Este método asegura que los datos procesados se conserven de manera estructurada y accesible, facilitando su uso en análisis posteriores o para la verificación de resultados.

4.1.6 FUNCIÓN GUARDAR IMAGEN HIGHLIGHTED

Esta función tiene como objetivo **guardar** una **imagen resaltada** en un archivo **BMP**. Esta imagen resaltada contiene los píxeles de interés identificados en una matriz de valores, y su almacenamiento en formato **BMP** permite una fácil visualización y análisis.

El proceso comienza leyendo la imagen original desde la ruta especificada por ``rutaImagenOriginal`` utilizando el método ``ReadBMP`` de la clase ``C_Image``. Luego, se determina el número de filas y columnas de la imagen.

A continuación, la función recorre cada píxel de la imagen. Si el valor del píxel en la matriz ``highlighted`` no es **246** (indicativo de **sangre hiperdensa**), el píxel correspondiente en la imagen es **establecido a negro (valor 0)**. Esto resalta los píxeles de interés en la imagen final.



```
void GuardarImagenHighlighted(const std::string& rutaImagenOriginal, const std::vector<std::vector<int>>& highlighted, const std::string& rutaArchivoSalida) {
    std::cerr << "Guardando imagen highlighted en archivo BMP...\n";
    C_Image imagenHighlighted;
    imagenHighlighted.ReadBMP(rutaImagenOriginal.c_str());

    long filas = imagenHighlighted.LastRow() + 1;
    long columnas = imagenHighlighted.LastCol() + 1;

    for (long i = 0; i < filas; ++i) {
        for (long j = 0; j < columnas; ++j) {
            if (highlighted[i][j] != 246) {
                imagenHighlighted(i, j) = 0;
            }
        }
    }

    imagenHighlighted.WriteBMP(rutaArchivoSalida.c_str());
    std::cerr << "Imagen guardada en archivo BMP.\n";
}
```

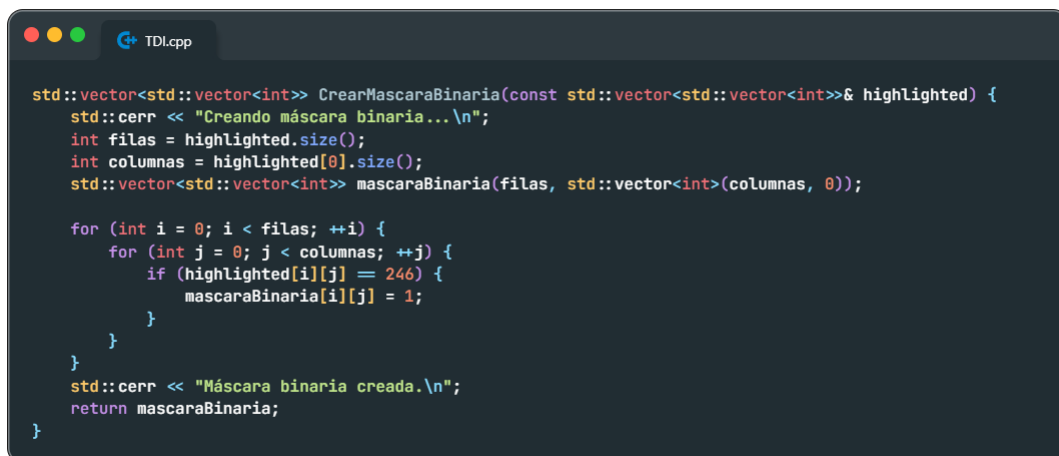
Figura 7: Función `GuardarImagenHighlighted`.

Finalmente, la imagen procesada se guarda en la ruta especificada por ``rutaArchivoSalida`` utilizando el método ``WriteBMP`` de la clase ``C_Image``. Este proceso asegura que los píxeles relevantes se destaquen en la imagen guardada, facilitando su revisión y análisis posterior.

4.1.7 FUNCIÓN CREAR MÁSCARA BINARIA

Esta función tiene como objetivo **generar una máscara binaria** a partir de una matriz resaltada, donde los **píxeles de interés** (indicados por el valor **246**) se marcan con un **1** y todos los **demás píxeles** se marcan con un **0**. Esta máscara binaria es crucial para el procesamiento posterior, como la detección de **contornos** y la **clasificación de hematomas** en imágenes de TAC craneal.

El proceso comienza determinando el número de filas y columnas de la matriz ``highlighted``. Se crea una nueva matriz ``mascaraBinaria`` de las mismas dimensiones, inicializada con ceros. Luego, la función recorre cada píxel de la matriz ``highlighted``.



```
std::vector<std::vector<int>> CrearMascaraBinaria(const std::vector<std::vector<int>>& highlighted) {
    std::cerr << "Creando máscara binaria...\n";
    int filas = highlighted.size();
    int columnas = highlighted[0].size();
    std::vector<std::vector<int>> mascaraBinaria(filas, std::vector<int>(columnas, 0));

    for (int i = 0; i < filas; ++i) {
        for (int j = 0; j < columnas; ++j) {
            if (highlighted[i][j] == 246) {
                mascaraBinaria[i][j] = 1;
            }
        }
    }

    std::cerr << "Máscara binaria creada.\n";
    return mascaraBinaria;
}
```

Figura 8: Función `CrearMascaraBinaria`.

Si el valor del píxel es **246**, se establece el valor correspondiente en la `mascaraBinaria` a 1.

Finalmente, la función imprime un mensaje indicando que la máscara binaria ha sido creada y devuelve la matriz `mascaraBinaria`. Esta máscara binaria simplifica el análisis de la imagen al reducir los datos a una representación binaria de los píxeles relevantes, facilitando así los pasos de procesamiento y análisis subsiguientes.

4.1.8 FUNCIÓN ENCONTRAR CONTORNOS

Como esta función es una de las más complejas, la trataremos a fondo por partes, cada una de las cuales está expuesta a continuación.

4.1.8.1 INICIALIZACIÓN Y PREPARACIÓN

Mensaje de inicio: Se imprime un mensaje indicando que el proceso de encontrar contornos ha comenzado.

Dimensiones de la matriz: Se obtienen el número de filas y columnas de la máscara binaria.

Matriz de visitados: Se inicializa una matriz del mismo tamaño que la máscara binaria para llevar un registro de los píxeles que ya han sido visitados.



```
std::cerr << "Encontrando contornos...\n";
int filas = mascaraBinaria.size();
int columnas = mascaraBinaria[0].size();
std::vector<std::vector<bool>> visitado(filas, std::vector<bool>(columnas, false));
```

Figura 9: Inicialización EncontrarContornos.

4.1.8.2 FUNCIONES LAMBDA PARA VALIDACIÓN Y DETECCIÓN DE BORDES

esValido: Verifica si un par de coordenadas (x, y) está dentro de los límites de la matriz.

esBorde: Determina si un píxel es un borde. Un píxel es considerado borde si tiene valor 1 y al menos uno de sus vecinos inmediatos (arriba, abajo, izquierda, derecha) tiene valor 0.



```
auto esValido = [&](int x, int y) {
    return x >= 0 && x < filas && y >= 0 && y < columnas;
};

auto esBorde = [&](int x, int y) {
    if (!esValido(x, y) || mascaraBinaria[x][y] == 0) {
        return false;
    }
    static int dx[] = { 0, 1, 0, -1 };
    static int dy[] = { 1, 0, -1, 0 };
    for (int i = 0; i < 4; ++i) {
        int nx = x + dx[i], ny = y + dy[i];
        if (esValido(nx, ny) && mascaraBinaria[nx][ny] == 0) {
            return true;
        }
    }
    return false;
};
```

Figura 10: Funciones Lambda EncontrarContornos

4.1.8.3 BUCLE PRINCIPAL PARA ENCONTRAR CONTORNOS

Bucle doble para recorrer la matriz: Los bucles for recorren cada píxel de la máscara binaria.

Condición de borde no visitado: Si un píxel no ha sido visitado y es un borde, se inicia el proceso de seguimiento del contorno.

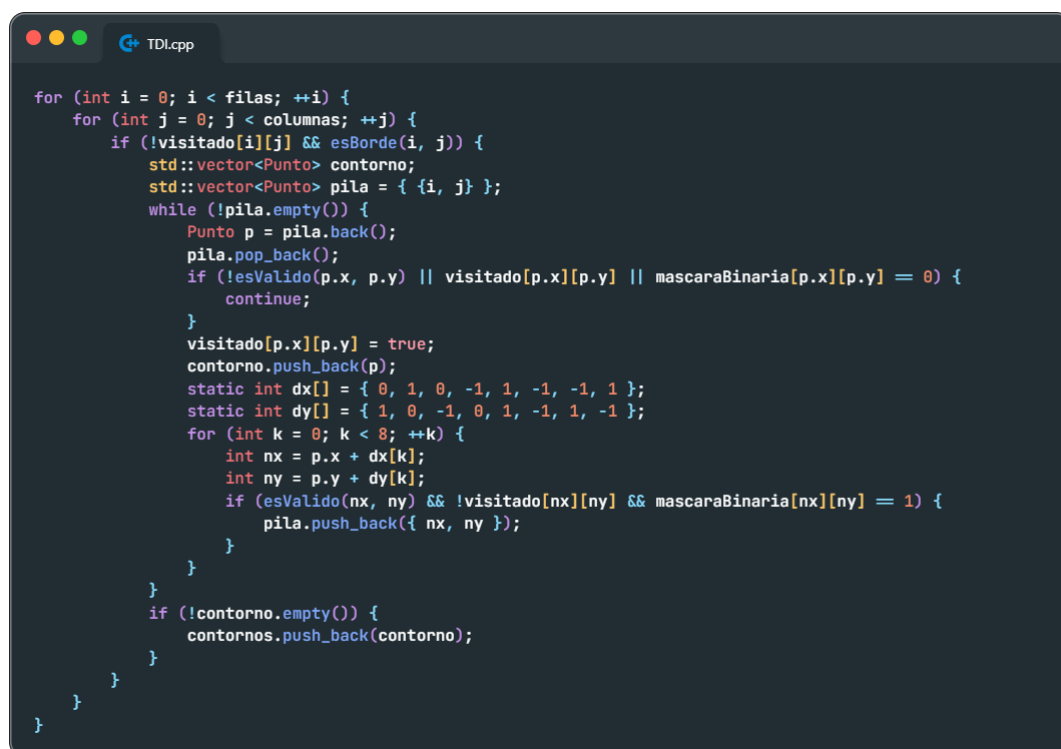
Inicialización del contorno: Se crea un vector para almacenar el contorno y una pila para el algoritmo de seguimiento.

Seguimiento del contorno: Mientras la pila no esté vacía, se extrae el último píxel de la pila y se verifica si es válido, no ha sido visitado, y es parte del contorno (valor 1 en la máscara binaria).

Marcado como visitado: El píxel se marca como visitado y se añade al vector de contorno.

Vecinos del píxel: Se examinan los vecinos del píxel en todas las direcciones (incluyendo diagonales). Si un vecino es válido, no ha sido visitado, y es parte del contorno, se añade a la pila para su posterior procesamiento.

Almacenamiento del contorno: Una vez procesado un contorno completo, se añade al vector de contornos.



```
TDL.cpp

for (int i = 0; i < filas; ++i) {
    for (int j = 0; j < columnas; ++j) {
        if (!visitado[i][j] && esBorde(i, j)) {
            std::vector<Punto> contorno;
            std::vector<Punto> pila = { {i, j} };
            while (!pila.empty()) {
                Punto p = pila.back();
                pila.pop_back();
                if (!esValido(p.x, p.y) || visitado[p.x][p.y] || mascaraBinaria[p.x][p.y] == 0) {
                    continue;
                }
                visitado[p.x][p.y] = true;
                contorno.push_back(p);
                static int dx[] = { 0, 1, 0, -1, 1, -1, -1, 1 };
                static int dy[] = { 1, 0, -1, 0, 1, -1, 1, -1 };
                for (int k = 0; k < 8; ++k) {
                    int nx = p.x + dx[k];
                    int ny = p.y + dy[k];
                    if (esValido(nx, ny) && !visitado[nx][ny] && mascaraBinaria[nx][ny] == 1) {
                        pila.push_back({ nx, ny });
                    }
                }
            }
            if (!contorno.empty()) {
                contornos.push_back(contorno);
            }
        }
    }
}
```

Figura 11: Bucle para encontrar contornos de la función *EncontrarContornos*.

4.1.9 FUNCIÓN CLASIFICAR HEMATOMA

Esta función tiene como objetivo **clasificar un hematoma** en función de la **forma** de sus **contornos** previamente identificados en una imagen de **TAC** craneal. Esta clasificación se basa en la **relación de aspecto** de los contornos, que puede indicar diferentes tipos de hematomas, como hematomas **epidurales** o **subdurales**. A

continuación se explica detalladamente cómo funciona la función y para qué sirve cada parte del código.

La función comienza imprimiendo un mensaje en la consola para indicar que el proceso de clasificación del hematoma ha comenzado. Luego, se recorre cada contorno en el vector de contornos proporcionado como argumento. Para cada contorno, se inicializan cuatro variables (**minX**, **minY**, **maxX** y **maxY**) que se utilizan para determinar los límites del contorno en términos de sus **coordenadas mínimas y máximas** en los **ejes x e y**.

Dentro del bucle, se recorre cada punto del contorno para actualizar estos límites. Se **comparan** las **coordenadas x e y** de cada punto **con las variables** **minX**, **minY**, **maxX** y **maxY**, y se actualizan en consecuencia. Este proceso permite identificar la caja delimitadora más pequeña que contiene todos los puntos del contorno, lo cual es crucial para calcular las dimensiones del contorno.



```
std::string ClasificarHematoma(const std::vector<std::vector<Punto>>& contornos) {
    std::cerr << "Clasificando hematoma...\n";
    for (const auto& contorno : contornos) {
        int minX = INT_MAX, minY = INT_MAX, maxX = INT_MIN, maxY = INT_MIN;
        for (const auto& punto : contorno) {
            minX = std::min(minX, punto.x);
            minY = std::min(minY, punto.y);
            maxX = std::max(maxX, punto.x);
            maxY = std::max(maxY, punto.y);
        }
        double ancho = maxX - minX + 1;
        double altura = maxY - minY + 1;
        double relacionAspecto = ancho / altura;

        if (relacionAspecto > 1.0 && relacionAspecto < 2.0) {
            return "Hematoma Epidural (Forma de Lente Biconvexa)";
        }
        else if (relacionAspecto >= 2.0) {
            return "Hematoma Subdural (Forma de Semiluna)";
        }
    }
    std::cerr << "Clasificación completada.\n";
    return "Hematoma Desconocido";
}
```

Figura 12: Función ClasificarHematoma.

Una vez **determinados** los límites del contorno, se **calcula** el **ancho** y la **altura** de la caja delimitadora **utilizando** las **diferencias** entre las **coordenadas máximas y mínimas** en los ejes x e y, respectivamente. **Se añade 1 a estas diferencias** para **incluir** tanto el **píxel inicial como el final en las dimensiones**. A continuación, se calcula la relación de aspecto dividiendo el ancho entre la altura.

La **relación de aspecto** es una medida **importante** que permite diferenciar entre diferentes tipos de hematomas. Si la relación de aspecto está **entre 1.0 y 2.0**, el contorno se clasifica como un **hematoma epidural**, que típicamente tiene una forma de **lente biconvexa**. Si la relación de aspecto es **mayor o igual a 2.0**, el contorno se clasifica como un **hematoma subdural**, que suele tener una forma de **semiluna**.

4.1.10 FUNCIÓN PROCESAR IMAGEN

La función ``ProcesarImagen`` es el núcleo del programa. Esta función comienza imprimiendo un mensaje en la consola para indicar que el proceso de procesamiento de la imagen ha comenzado. Luego, se crea un objeto ``C_Image`` y se lee la imagen de entrada en formato **BMP** desde la ruta especificada por ``rutaArchivoEntrada``. Esta imagen se utiliza como base para todos los pasos de procesamiento subsecuentes.

El siguiente paso es **crear** una **matriz resaltada** eliminando los cinco valores más blancos de la imagen, lo que ayuda a aislar las áreas relevantes para el análisis del hematoma. Esto se realiza mediante la función ``Resaltar246``, que devuelve una matriz donde se destacan los píxeles de interés. Luego, se genera una carpeta de salida única utilizando la función ``CrearCarpetaSalida``, que asegura que los datos de cada ejecución del programa se mantengan organizados y no se sobrescriban.

Una vez creada la carpeta de salida, se guarda una versión de la imagen donde se han eliminado los valores correspondientes al cráneo, destacando solo los píxeles de interés. Esta imagen se guarda en la ruta especificada por ``rutaArchivImagenSinCraneo``. A continuación, se eliminan los píxeles con baja densidad en la matriz resaltada mediante la función ``EliminarPíxelesBajaDensidad``, lo cual es crucial para reducir el ruido y enfocarse en las áreas con mayor densidad de píxeles relevantes, como los hematomas.

Para mejorar aún más la calidad de la imagen, se aplica un **filtro de mediana** a la matriz de píxeles densos utilizando la función ``AplicarFiltroMediana``. Este paso **suaviza** la **imagen** y mejora la identificación del área afectada, **eliminando el ruido residual** y resaltando mejor las características importantes. La matriz filtrada resultante se guarda en un archivo de texto, permitiendo una revisión y análisis posterior de los datos procesados.

Además de guardar la matriz filtrada, se guarda una versión **BMP** de la imagen filtrada en la carpeta de salida, proporcionando una representación visual de los píxeles relevantes después de la filtración. También se guarda la imagen original en la carpeta de salida para referencia y comparación con las versiones procesadas.

A continuación, se crea una **máscara binaria** a partir de la matriz filtrada utilizando la función ``CrearMascaraBinaria``, y se identifican los contornos de las áreas de interés mediante la función ``EncontrarContornos``. Estos contornos son esenciales para la clasificación del hematoma.

La clasificación del hematoma se realiza mediante la función ``ClasificarHematoma``, que analiza los contornos encontrados para determinar el tipo de hematoma (por ejemplo, epidural o subdural). El tipo de hematoma identificado se guarda en un archivo de texto dentro de la carpeta de salida. Si hay un error al abrir el archivo, se imprime un mensaje de error en la consola.

Finalmente, la función imprime un mensaje indicando que el proceso de análisis de la imagen se ha completado y muestra el tipo de hematoma identificado. En resumen, la función ``ProcesarImagen`` coordina todas las etapas del análisis de imágenes de **TAC** craneal, desde la lectura y procesamiento de la imagen hasta la clasificación y almacenamiento de los resultados, asegurando una gestión eficiente y precisa de los datos.

```

void ProcesarImagen(const std::string& rutaArchivoEntrada) {
    std::cerr << "Procesando imagen...\n";
    // Crear un objeto imagen y leer el archivo BMP de entrada
    C_Imagen imagen;
    imagen.ReadBMP(rutaArchivoEntrada.c_str());

    // Crear la matriz highlighted eliminando los cinco valores más blancos
    std::vector<std::vector<int>> highlighted = Resaltar246(imagen);

    // Crear la carpeta de salida
    std::string carpetaSalida = CrearCarpetaSalida();

    // Guardar la imagen sin cráneo en un archivo BMP
    std::string rutaArchivoImagenSinCraneo = carpetaSalida + "/sin_craneo.bmp";
    GuardarImagenSinCraneo(rutaArchivoEntrada, highlighted, rutaArchivoImagenSinCraneo);

    // Eliminar los píxeles con baja densidad
    std::vector<std::vector<int>> highlightedDensa = EliminarPíxelesBajaDensidad(highlighted, 9, 8);

    // Aplicar el filtro de mediana
    std::vector<std::vector<int>> highlightedFiltrada = AplicarFiltroMediana(highlightedDensa);

    // Guardar la matriz highlighted filtrada en un archivo de texto
    std::string rutaArchivoMatrizHighlighted = carpetaSalida + "/highlighted_Matriz_Filtrada.txt";
    GuardarMatrizHighlighted(highlightedFiltrada, rutaArchivoMatrizHighlighted);

    // Guardar la imagen highlighted filtrada en un archivo BMP
    std::string rutaArchivoImagenHighlighted = carpetaSalida + "/highlighted_Imagen_Filtrada.bmp";
    GuardarImagenHighlighted(rutaArchivoEntrada, highlightedFiltrada, rutaArchivoImagenHighlighted);

    // Guardar la imagen original en un archivo BMP en la carpeta de salida
    std::string rutaArchivoImagenOriginal = carpetaSalida + "/original_Imagen.bmp";
    imagen.WriteBMP(rutaArchivoImagenOriginal.c_str());

    // Crear máscara binaria y encontrar contornos
    std::vector<std::vector<int>> mascaraBinaria = CrearMascaraBinaria(highlightedFiltrada);
    std::vector<std::vector<Punto>> contornos;
    EncontrarContornos(mascaraBinaria, contornos);

    // Clasificar el hematoma basado en los contornos
    std::string tipoHematoma = ClasificarHematoma(contornos);

    // Guardar el tipo de hematoma en un archivo de texto
    std::string rutaArchivoTipoHematoma = carpetaSalida + "/tipo_Hematoma.txt";
    std::ofstream outFile(rutaArchivoTipoHematoma);
    if (outFile) {
        outFile << tipoHematoma << std::endl;
        outFile.close();
    }
    else {
        std::cerr << "Error al abrir el archivo de salida: " << rutaArchivoTipoHematoma << std::endl;
    }

    std::cerr << "Proceso completado. Tipo de hematoma: " << tipoHematoma << std::endl;
}

```

Figura 13: Función ProcesarImagen.

4.1.11 FUNCIÓN MAIN

Esta función en el programa, solo llama al procedimiento principal 'ProcesarImagen' y le pasa como argumento la ruta al archivo **BMP** de entrada.

```

int main(int argc, char* argv[]) {
    if (argc < 2) {
        std::cerr << "Uso: " << argv[0] << " <ruta al archivo BMP>" << std::endl;
        return 1;
    }

    std::string rutaArchivoEntrada = argv[1];

    // Procesar la imagen
    ProcesarImagen(rutaArchivoEntrada);

    return 0;
}

```

Figura 14: Función Main.

5 ESTUDIO DE LA IMPLEMENTACIÓN

6 CONCLUSIONES

7 BIBLIOGRAFÍA

Dalbayrak, S., Gumustas, S., Bal, A., & Akansel, G. (2011). *Early and delayed CT findings in patients with mild-to-moderate head trauma*. Turkish neurosurgery, 21(4), 591–598.

Albers, C. E., von Allmen, M., Evangelopoulos, D. S., Zisakis, A. K., Zimmermann, H., & Exadaktylos, A. K. (2013). *What is the incidence of intracranial bleeding in patients with mild traumatic brain injury? A retrospective study in 3088 Canadian CT head rule patients*. BioMed research international, 2013, 453978. <https://doi.org/10.1155/2013/453978>

Chicote Álvarez, E., González Castro, A., Ortiz Lasa, M., Jiménez Alfonso, A., Escudero Acha, P., Rodríguez Borregán, J. C., Peñasco Martín, Y., & Dierssen Sotos, T. (2018). Epidemiología del traumatismo craneoencefálico en la población mayor de 65 años a lo largo de 25 años. *Revista española de anestesiología y reanimación*, 65(10), 546–551. <https://doi.org/10.1016/j.redar.2018.06.003>