UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TREBALL FI DE GRAU

Grau en Enginyeria Biomèdica

LLIBRERIA D'APLICACIONS DE PROCESSAMENT D'IMATGES BIOMÈDIQUES AMB ENTORN GRÀFIC D'USUARI

Memòria i Annexos

Autor: Andrea Blasi Núñez

Director: Raúl Benítez Iglesias

Convocatòria: Addicional. Quadrimestre de tardor 2019-2020.

Resum

Aquest projecte és la continuació d'un treball consistent en el desenvolupament d'un software de processat d'imatges mèdiques amb interfície gràfica d'usuari (GUI). L'objectiu d'aquesta tesi és l'actualització tant del software com de la GUI amb la finalitat de complementar les funcionalitats ja existents i d'oferir una eina més propera i intuïtiva al personal mèdic.

En lo relatiu al software, l'aplicació ha triomfat en la implementació dels nous mètodes de segmentació proposats, que han permès una segmentació ràpida i precisa. En concret l'aprofundiment en la segmentació per Barreja de Gaussianes ha aportat una diferència significativa en comparació amb les antigues metodologies. Tots els mètodes i funcions han estat optimitzats mitjançant la simplificació del codi, disminuint així el temps d'execució. En addició, s'han implementat una sèrie de funcionalitats entre les que destaquen la visualització de l'histograma de la imatge d'entrada, la possibilitat de recuperar la imatge original mitjançant la eliminació del processat, l'accessibilitat a informació relativa a restriccions, limitacions i instruccions de l'aplicació i l'habilitat de guardar la informació i els resultats obtinguts.

Entre les actualitzacions de la GUI es troben la capacitat de canviar la llengua de l'aplicació (català, castellà i anglès), el redisseny de la pantalla principal per fer-la més atraient i intuïtiva, la millora de la qualitat de visualització de les imatges i, per últim, la incorporació d'un ampli apartat d'ajuda tant del funcionament i metodologia de cada funcionalitat de l'aplicació com dels conceptes que en ella es tracten.

Resumen

Este proyecto es la continuación de un trabajo consistente en el desarrollo de un software de procesado de imágenes médicas con interfaz gráfica de usuario (GUI). El objetivo de esta tesis es la actualización tanto del software como de la GUI con el fin de complementar las funcionalidades ya existentes y de ofrecer una herramienta más cercana e intuitiva al personal médico.

En lo relativo al software, la aplicación ha triunfado en la implementación de los nuevos métodos de segmentación propuestos, que han permitido una segmentación rápida y precisa. En concreto la profundización en la segmentación por Mezcla de Gaussianas ha aportado una diferencia significativa en comparación con las antiguas metodologías. Todos los métodos y funciones han sido optimizados mediante la simplificación del código, disminuyendo así el tiempo de ejecución. En adición, se han implementado una serie de funcionalidades entre las que destacan la visualización del histograma de la imagen de entrada, la posibilidad de recuperar la imagen original mediante la eliminación del procesado, la accesibilidad a información relativa a restricciones, limitaciones e instrucciones de la aplicación y la habilidad de guardar la información y los resultados obtenidos.

Entre las actualizaciones de la GUI se encuentran la capacidad de cambiar la lengua de la aplicación (catalán, castellano e inglés), el rediseño de la pantalla principal para hacerla más atrayente e intuitiva, la mejora de la calidad de visualización de las imágenes y, por último, la incorporación de un amplio apartado de ayuda tanto del funcionamiento y metodología de cada funcionalidad de la aplicación como de los conceptos que en ella se tratan.

Abstract

This project is the continuation of another thesis consisting of a medical image processing software through the development of a graphical user interface (GUI). Thus, the aim of this thesis is to update both the software and the GUI in order to complement the existing functionalities and to offer a closer and more intuitive approach to hospitals and doctors.

In terms of the software, the application has succeeded in implementing the proposed new segmentation methods, which have allowed a faster and more accurate processing. Specially, the deep understanding of the segmentation through the Gaussian Mixture Model has made a significant effect on segmentation efficiency. All methods and functions have been optimized by simplifying the code to the full extent and therefore the execution time has been reduced. In addition, many functionalities have been implemented, the outstanding ones are the display of the input image's histogram, the possibility of undo or reverse the processing, the accessibility to information regarding restrictions, limitations and instructions and the ability to save the information and results obtained.

The GUI's updates include the ability to change the application language (all Catalan, Spanish and English are available), the redesign of the main screen so as to gain attractiveness and make it more intuitive, the improvement of the displayed images' quality and, finally, the incorporation of a help section regarding not only the methodology of each functionality of the application but also every concept that is addressed.

Agraïments

En primer lloc voldria agrair tots els esforços del meu tutor de projecte, el Raúl Benítez, per dirigir-me sempre en la bona direcció i per motivar-me a continuar aprenent i a superar els meus límits.

El camí fàcil hagués estat desenvolupar un projecte en Matlab, programa que coneixia i amb el que tenia molta experiència. Però gràcies al Raúl vaig trobar la motivació i el temps necessari per arriscar-me a aprendre no només un llenguatge de programació gairebé des de zero, sinó tot un concepte com és la interfície gràfica d'usuari, activitat que he gaudit fins l'últim moment i que tant útil em serà en un futur.

També m'agradaria donar les gràcies a tots els programadors i programadores que es prenen el temps necessari per resoldre els dubtes als fòrums online dels nouvinguts al món de la programació, que expliquen conceptes en un llenguatge senzill i accessible a tothom i que distribueixen lliurement el codi que tant d'esforç els hi ha costat. Ells fan possible que persones com jo aprenguin cada dia, formant a les noves generacions i conseqüentment, contribuint al progrés.

Per últim, voldria agrair el suport de la meva família, que tant orgullosa està de la meva carrera professional fins i tot abans de que aquesta comenci. En especial, dono les gràcies a la meva àvia, que m'ha acollit en aquests moments tant difícils i que mai ha dubtat en escoltar-me i donar-me suport a cada dificultat que se'm plantejava, tot i no entendre una paraula del que li explicava.

Glossari

csv: Format d'arxiu contenidor de taules on els valors estan separats per coma.

jpg: Format gràfic basat en un algoritme de compressió d'imatges.

Matplotlib: Llibreria de Python per a dibuixar imatges i gràfics de qualitat en 2D.

OOP: Programació orientada a objectes.

QImage: Format d'imatge compatible amb la representació mitjançant QtWidgets.

png: Format gràfic basat en un algoritme de compressió d'imatges.

Pop-ups: Finestres emergents.

QtDesigner: Programa d'assistència al disseny de interfícies d'usuari propi de Python i compatible amb QtWidgets.

QtWidgets: eines de Python per tot tipus de visualitzacions (imatges, botons, accions, etc.) en interfícies d'usuari.

Python: Llenguatge de programació orientada a objectes.

PyQt: Binding de la biblioteca gràfica Qt de C++ per a Python.

Script: Arxiu d'ordres escrites en un llenguatge de programació.

Seaborn: Llibreria de Python per a la representació de dades estadístiques amb estils diferents.

Skimage: abreviació de scikit-image, llibreria per al processament d'imatge en Python.

TFG: Treball Final de Grau.

tif: Format gràfic basat en un algoritme de compressió d'imatges

ToolBox: set de funcions o eines de software accesibles desde un únic menú.

Tooltip: missatge descriptiu que apareix en pasar el cursor per sobre dels botons de la interfície d'usuari.

UML: llenguatge unificat de modelització.

Warnings: Avís.

Índex

Índex de taules

Índex d'equacions

Índex de figures

1. Prefaci

1.2. Origen del treball

Aquest treball es basa en la continuació del projecte de fi de grau d'un altre estudiant d'enginyeria biomèdica a la facultat d'Enginyeria de Barcelona Est, la Sandra Bernaus. L'objectiu inicial era col·laborar amb l'Hospital Sant Joan de Déu per proporcionar una plataforma que servís com a eina de suport per l'estudi de malalties rares que pogués ser utilitzada en els ordinadors de l'hospital [1]. La falta de temps no va fer possible l'assoliment d'una aplicació estrictament completa, però va servir com a base sobre la qual s'ha treballat en aquest projecte, tot implementant una millora i extensió de les funcionalitats de la mateixa.

1.3. Motivació

Des de la reconstrucció d'imatges de ressonància magnètica nuclear fins a l'extracció d'estructures anatòmiques cerebrals i el desenvolupament d'estudis poblacionals, el camp de tractament d'imatges mèdiques ha experimentat un creixement exponencial tant en el sector públic com en el sector privat. Avui en dia, gràcies als avançats escàners i programari de reconstrucció d'imatges és possible la identificació d'òrgans i teixits, així com l'obtenció de dades que ajuden a caracteritzar i quantificar les patologies [2]. És per això que la importància de tenir a disposició imatges fiables per tal de realitzar processos ràpids de diagnòstic amb alt grau de certesa és vital.

L'èxit d'un diagnòstic clínic basat en imatges depèn de l'exactitud amb la qual el professional de la medicina pugui visualitzar l'objecte d'estudi [3]. Cada cop és més comú que aquests facultatius demanin eines que els permetin localitzar òrgans i teixits amb major precisió i rapidesa, així com la identificació i caracterització quantitativa de les patologies presents en ells, per tal de fer un millor diagnòstic. No obstant, el processament i anàlisi d'imatges mèdiques és un problema complex que requereix d'un coneixement especialitzat, per això és usual que els metges tinguin dificultats per interpretar els processaments i la metodologia que s'ha seguit. També és comú que els encarregats de processar les imatges requereixin d'assessorament per part de l'expert per prendre les decisions pertinents pel que fa a l'ús de la compressió d'imatges; en efecte, en un intent d'estalviar emmagatzematge es pot perdre qualitat de la mateixa a l'hora d'emetre un diagnòstic.

És per aquesta raó que el contacte constant entre experts en processament d'imatges i els professionals de la medicina és indispensable, així com el mantenir-se al corrent de les noves actualitzacions en ambdós camps. Per tant, el motiu principal pel que s'ha impulsat aquest projecte

ha estat la creació d'una aplicació de processat d'imatge que fos accessible, manejable i intuïtiva pels professionals de la medicina, per tal de construir un canal de comunicació directa entre l'àmbit tecnològic i biomèdic.

En concret, l'aplicació tracta d'ajudar al reconeixement d'objectes pel diagnòstic de malalties mitjançant diferents funcionalitats com poden ser la visualització d'imatges, l'emmagatzematge de les mateixes, la millora de la qualitat i el contrast i les tasques de processament, segmentació i extracció d'informació amb la finalitat de dur a terme una mesura i quantificació de paràmetres anatòmics i fisiològics. S'ha optat per fer proves amb imatges mèdiques de tot tipus de provinences, abastant totes les branques de la medicina i deixant una gran varietat de funcionalitats a l'abast de l'usuari per a una possible futura especialització.

1.4. Requeriments previs

Els algoritmes numèrics que componen el nucli d'aquesta tesi (que es detallen en el capítol de disseny) han estat desenvolupats íntegrament en Python (versió 3.7.4). Per tant, es precisa de la instal·lació de Python i QtDesginer per a la realització del projecte. També d'una sèrie de llibreries específiques, entre les que es destaquen Scikit imatge, PyQt5, Matplotlib, scipy, datetime, NumPy, Pandas i Pyqtgraph. Totes elles són necessàries pel tractament i visualització d'imatges i disseny de la GUI.

Altres programes han sigut també utilitzats per a la redacció de la memòria, el disseny de diagrames UML, la planificació i el control de versions i servei de backup. En són exemples el Microsoft Word, Visual Paradigm, TeamGantt i Google Drive, respectivament.

Per últim, també s'ha precisat de comunicació directa entre l'Hospital Sant Joan de Déu i el tutor del projecte per tal de permetre el disseny d'un model de requeriment de l'aplicació, tot definint funcionalitats, disseny gràfic i limitacions.

2. Introducció

2.1. Objectius del treball

L'objectiu principal que aborda aquesta memòria és el de proposar, desenvolupar i validar una actualització tant del software com de la interfície gràfica d'usuari del projecte dut a terme per la Sandra Bernaus [1] per tal d'aconseguir una aplicació de visualització i processament d'imatges més eficient, completa i intuïtiva. L'aplicació té com a finalitat servir com a eina de suport per a professionals de la medicina i familiaritzar-los amb les metodologies de segmentació més sofisticades.

Hi ha dos enfocaments clarament diferenciats; les actualitzacions del software i les de la interfície d'usuari. Mentre que la primera es basa en la optimització del codi i la complementació de funcionalitats, la segona rau en una millora del disseny visual per a establir una connexió més directa amb l'usuari i la incorporació d'instruccions per esclarir les funcionalitats.

A la vegada, es desitja que l'aplicació permeti introduir, processar, visualitzar i extreure biomarcadors de les imatges per ajudar al diagnòstic de malalties. Per assolir aquest objectiu general es defineixen una sèrie d'objectius específics, classificats segons la seva relació amb el software o la interfície d'usuari:

SOFTWARE	INTERFÍCIE D'USUARI (GUI)
Completar les tècniques de segmentació amb unes de més complexes.	Dirigir l'aplicació a un ventall més ampli d'usuaris, fent possible l'aplicació en diferents idiomes.
Permetre a l'usuari definir paràmetres de la segmentació, per augmentar la capacitat d'adaptació a les seves necessitats.	Realitzar un contingut més enfocat als potencials usuaris de l'aplicació.
A causa de la variabilitat entre els diferents tipus d'imatge biomèdica, sempre tenir com a prioritat la adaptabilitat a les condicions de cadascuna mitjançant la interacció entre el programador i l'investigador	Actualitzar el disseny gràfic de la pantalla principal per fer-lo més atraient.

Simplificar i evitar redundàncies al codi per disminuir el temps d'execució.	Redissenyar els widgets de l'aplicació per fer-los més propers i familiars al usuari.
Optimitzar les funcions ja existents per fer-les més senzilles i robustes.	Millorar la qualitat de les imatges en la seva visualització.
Afegir eines complementàries: • Visualització de l'histograma. • Possibilitat d'eliminar el processat actual un cop realitzat. • Captura de la pantalla de l'aplicació. • Redirecció directa a impressió de les dades en paper. • Finestres emergents amb informació de restriccions i instruccions a seguir.	Afegir *Warnings* en cas que l'usuari en faci un ús erroni del programa i redirigir-lo al funcionament correcte. Orientar a l'usuari mitjançant la incorporació d'instruccions i ajudes.

Taula 1: Objectius específics del projecte relatius a l'actualització del software i la interfície d'usuari.

2.2. Estat previ al projecte

Per a una correcta comprensió del projecte s'ha de destacar el fet que aquest sigui la continuació del treball de fi de grau d'una altre estudiant, la Sandra Bernaus. Durant la realització de la memòria s'ha enfocat el projecte de dues maneres; mentre que hi ha punts, com per exemple els càlculs de despeses econòmiques i impacte mediambiental i social entre d'altres, en que s'ha considerat el projecte de la Sandra i el meu com un de sol, en la majoria de capítols s'ha optat per un enfocament comparatiu entre els dos.

Pels càlculs de despeses, impacte i temps total necessari s'ha trobat més lògic tractar les dues parts del projecte com a una sola per tal de poder comptabilitzar tots els efectes del projecte al complet, des de zero. En contraposició, per a poder fer una descripció adient de totes les actualitzacions s'ha considerat més apropiat explicar concretament què ha canviat respecte al projecte de la Sandra i per què. D'aquesta manera s'ha atorgat al projecte continuïtat i sentit en tots els apartats.

És per aquest motiu pel qual durant la memòria es fa referència a termes com ***primer i segon període o etapa*** amb molta freqüència. El primer període doncs, és el projecte de fi de grau de la Sandra Bernaus, mentre que el segon període fa referència a totes les actualitzacions que jo mateixa he dut a terme en aquest treball.

3. Marc teòric

El marc teòric d'aquest projecte està orientat a conèixer i comprendre l'estat de l'art de les aplicacions de processat d'imatges mèdiques i els conceptes que es tracten durant la memòria. Donat el fet que el projecte desenvolupa una interfície gràfica d'usuaris (GUI) amb processat d'imatges com a funcionalitat principal de l'aplicació, s'explicaran els conceptes pertinents a ambdós camps.

En primer lloc es descriuran les interfícies d'usuari i el llenguatge de programació orientat a objectes, conjuntament amb el motiu de l'elecció de Python en concret.

En segon lloc, al capítol de processat d'imatges, s'explicarà tot lo relatiu a les segmentacions que s'han implementat. Es començarà tractant els conceptes previs com poden ser soroll, filtres, histograma i overfitting. Seguidament es parlarà del concepte de segmentació d'imatges, tot comparant les diferents categories. Per últim es donaran breus descripcions des del punt de vista estadístic, entre els que trobem els conceptes de funció de densitat de probabilitat, distribucions gaussianes, algorismes de maximització d'expectatives i BIC.

3.1. Programació orientada a objectes

La programació orientada a objectes (*Object Oriented programming, OOP)* és un paradigma de la programació que utilitza unitats anomenades classes per a la organització del codi i objectes com a elements fonamentals [4] [5]. La avantatge principal d'aquest tipus de programació és la estreta relació amb la realitat, ja que els seus elements tracten d'emular el comportament i les relacions existents a la vida real. Així, un objecte és una abstracció d'algun fet o entitat que es troba al món real. Aquest consta d'atributs que representen les seves característiques i de mètodes que emulen el seu comportament. Les propietats i mètodes comuns a un conjunt d'objectes s'agrupen en classes. Aquestes esdevenen aleshores plantilles o prototips per crear nous objectes (instàncies) que segueixen el mateix comportament. Cada objecte però, pot tenir els seus mètodes i atributs particulars simultàniament amb els de la classe [6]. Un altre fet destacable és la capacitat d'heretar mètodes i atributs d'una classe a una altra, estalviant una gran quantitat de codi i de memòria i optimitzant la velocitat de compilació.

El llenguatge basat en programació orientada a objectes va ser escollit per programar l'aplicació arran dels múltiples beneficis que comporta, entre els que es destaquen la capacitat de re-utilitzar el codi mitjançant la instanciació de classes i l'herència, la senzillesa a l'hora d'abstreure el problema gràcies a un codi més intuïtiu, la facilitat d'afegir, suprimir o modificar nous objectes i la competència d'aïllar seccions d'un problema per provar-les independentment [7].

3.1.1. Python

Python és una programació informàtica de codi obert de propòsit general. Va ser dissenyat per expressar de forma clara i directa les instruccions que els programes han de seguir sense la necessitat d'indicar els detalls de baix nivell com els tipus de variables, la mida de les estructures de dades o el maneig de la memòria [8]. A part de ser fàcil al ús, la seva sintaxi és més simple que altres llenguatges. Al 2018, la revista *IEEE Spectrum* el va declarar guanyador en el cinquè rànquing interactiu anual dels principals llenguatges de programació [9].

S'han fet estudis per comparar els diferents llenguatges de programació orientada a objectes, n'és un exemple l'estudi dut a terme per L. Prechelt de la Universitat de Karlsuhe, Alemanya [8], qui va demostrar que dissenyar un programa amb Python comporta no més de la meitat del temps que fent-ho amb C, C++ o Java, sense comportar un decreixement de la fiabilitat d'aquest. A més, el programa resultant és la meitat de llarg.

Python ofereix una sèrie de llibreries amb eines específiques pel processat d'imatge, en concret s'ha fet ús de Scikit-Image. Aquest es composa d'una col·lecció d'algorismes implementats per una comunitat activa de persones voluntàries que està disponible sota la llicència liberal BSD Open Source [10]. Van der Walt et at. [10] justifica la utilització de la llibreria Scikit-Image afirmant que *la popularitat creixent de Python com a llenguatge de programació científica, juntament amb la creixent disponibilitat d'un gran ecosistema d'eines complementàries, el converteixen en un entorn ideal on produir un kit d'eines de processament d'imatges.*

3.2. Interfícies gràfiques d'usuari (GUI)

Durant els darrers 15 anys s'ha produït un increment en l'ús de la tecnologia per part d'usuaris que no estan formats en informàtica [11]. Això ha suposat una transició del disseny centrat en tecnologia al disseny centrat en l'usuari. Per definició, el disseny centrat en l'usuari es centra en les seves característiques, tasques i entorn per proporcionar una aplicació que compleixi els denominats requisits de l'usuari [12].

La interfície gràfica d'usuari (*Graphical User Interface, GUI*) es descriu com un programa informàtic que actua com a interfície d'usuari, on s'utilitzen un conjunt d'imatges i objectes gràfics per a representar la informació i les accions disponibles a la interfície [13]. El seu objectiu principal és el de proporcionar un entorn visual senzill per permetre una comunicació entre l'usuari i l'ordinador, fent coincidir la visualització de la GUI amb la semàntica o els models de l'usuari, per tal de fer-lo eficient en termes de facilitat d'ús i d'aprenentatge i resistència a l'error. D'aquesta manera la GUI es considera l'artefacte tecnològic d'un sistema interactiu que fa possible una interacció senzilla amb un sistema informàtic a

través de l'ús del llenguatge visual [14]. La manipulació directa del programa és una de propietats més estimades pels usuaris [15] i avui en dia es considera una característica de disseny estàndard per a aplicacions de programari. El primer cas de manipulació directa es va donar amb Sutherland's Sketchpad, els quals van crear un programa de disseny gràfic al 1963 [16].

Les interfícies gràfiques d'usuari han representat un canvi fonamental en la manera en què els humans interactuen amb els ordinadors. La diferència més radical ha estat la forma en que es representa la informació per pantalla. Així, en lloc d'utilitzar un llenguatge d'ordres altament especialitzat, la interfície gràfica d'usuari té la capacitat de representar la informació com a objectes o imatges visuals [17]. Aquests objectes es poden descriure en termes de la seva ubicació a la pantalla en relació amb les vores d'aquesta, amb altres objectes i en relació amb el punter del ratolí.

La dificultat de codificar interfícies tradicionals era molt més elevada, ja que no hi havia més remei que aprendre les ordres i els procediments del teclat per a cada aplicació utilitzada [17]. Sovint, la interacció amb l'ordinador precisava d'un diàleg tediós de sol·licituds i ordres. Entre els avantatges de la utilització d'interfícies gràfiques d'usuari destaquen la comoditat, el cost de desenvolupament, documentació i suport reduït, així com el fet que un cop s'aconsegueix una interfície estandarditzada no s'ha de començar de zero al desenvolupar una nova aplicació.

Es diu que les interfícies gràfiques d'usuari han arribat a quasi substituir a les de text. Les anomenades interfícies basades en text (*Text User Interface, TUI*) són un tipus d'interfície que utilitza únicament text ASCII. Molts estudis donen suport a un canvi de TUI a GUI, tot mostrant els seus avantatges. Rauterberg va dur a terme una avaluació de la selecció de menús mitjançant una TUI i una GUI tant per a usuaris principiants com per experts. Els resultats van demostrar que fins i tot els experts necessitaven fins a un 51% menys de temps per realitzar les tasques [18]. Davis i Bostrom van fer un examen de les possibles diferències entre les dues durant la realització de tasques de directori de fitxers i estructura per trobar que existia una millora significativa en la habilitat dels usuaris novells per aprendre amb la interfície gràfica d'usuari [19]. Barker i Sloane, Inc. els van comparar tenint en compte les variables de número de tasques, precisió, frustració i cansament en els usuaris al dur a terme les activitats de tasques de processament de textos i fulls de càlcul. Va descobrir que els usuaris havien arribat a fer un 35% més de tasques un 17% més precises i amb un nivell de frustració i fatiga menor [20]. Per últim, Staggers els va comparar en funció del temps de resposta, els errors i la satisfacció per part del personal d'infermeria, amb uns resultats que demostraven la superioritat de GUI respecte TUI [21].

3.3. Processat d'imatges

3.3.1. Conceptes previs

Els conceptes relatius al processat d'imatges que es tracten a continuació s'han considerat claus per a l'enteniment de la memòria. Es començarà amb nocions tals com el soroll en imatges, ja que és la problemàtica principal en el processat d'imatges. Seguidament es descriuen i comparen els filtres passa baixa més representatius, ja que s'han considerat per l'aplicació. També l'histograma, sent una de les funcionalitats afegides en aquest segon període del projecte. I, per últim, la sobre-segmentació, ja que va representar el principal obstacle a l'hora d'implementar les segmentacions.

3.3.1.1. Soroll

Segons P. Patidar et al. el soroll d'una imatge es defineix com *la variació aleatòria de la informació de brillantor o de color a les imatges produïdes pel sensor i els circuits d'un escàner o càmera digital* [22]. Altres autors tals com A. K. Boyat estan d'acord amb Patidar al classificar els tipus de soroll en soroll Gaussià, soroll Salt-and-Pepper, soroll de Poisson i soroll *speckle.* El primer apareix quan un dels canals està més amplificat que els altres. El soroll Salt-and-Pepper tindrà píxels foscos a zones clares i viceversa. Aquest pot ser causat per píxels morts, errors en la conversió d'analògic a digital, etc. El soroll de Poisson és un tipus de soroll electrònic que es és produït en el cas que el nombre finit de partícules que transporten energia (electrons o fotons) del dispositiu òptic, sigui tan petit que produeixi fluctuacions estadístiques detectables en la mesura. Per últim, el soroll *Speckle* és un soroll granular que existeix inherentment, degradant la qualitat del radar.

A continuació es mostra un exemple de cadascun:

Figura 1: Comparació de l'efecte en una imatge dels diferents tipus de soroll. En la fila superior es troba en la posició esquerra la imatge original, al centre la mateixa imatge amb soroll Gaussià aplicat i al costat dret la imatge original amb soroll Salt-and-Pepper. A la fila inferior, en la posició esquerra es mostra la imatge original amb soroll de Poisson aplicat i finalment a la dreta la imatge original amb soroll Speckle. El soroll aplicat ha tingut en tots els casos una desviació estàndard de 0.025.[1]

3.3.1.2. Filtres passa baixa

En imatges, al igual que passa per les senyals, els filtres actuen al domini de la freqüència. La imatge es transforma del domini espacial al de la freqüència mitjançant un procés de convolució. Així, la terminologia dels filtres passa alt o passa baix deriva de la repercussió en el món de la freqüència. La alta freqüència espacial està associada a canvis freqüents de nivell de píxel, mentre que la baixa freqüència ho està a la uniformitat [23]. Els filtres passa baixa retenen les baixes freqüències i eliminen les altes, com poden ser vores, línies i alguns tipus de soroll. Els filtres més populars d'aquest tipus, i els que s'han utilitzat en el projecte són el filtre mitjana i el filtre gaussià.

[1] Aquestes imatges han sigut extretes de l'article de P. Patidar et al. [22].

- **Filtre mediana**

Aquest filtre substitueix el píxel que s'està analitzant per la mitjana del nivell d'intensitat de gris respecte als píxels del voltant [24]. Vegeu el següent exemple:

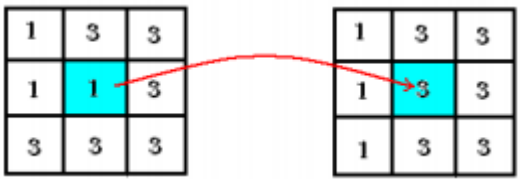

Figura 2: Filtre mitjana amb una màscara de 3x3.[1]

- **Filtre gaussià**

El tipus més comú de filtre Gaussià és el Kernel, el qual té una màscara de 3x3 que assigna els coeficients amb més pes al píxel central. A la resta de píxels propers un valor inferior i quan els píxels estan més allunyats encara menor [24].

$$W = \frac{1}{16} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Figura 3: màscara Kernel corresponent a un filtre Gaussià.[2]

3.3.1.3. Histograma

L'histograma d'una imatge es pot definir com la representació compacta de la distribució de nivells de gris [25]. La metodologia es basa en comptar quants píxels hi ha per a la imatge per cada nivell de píxel, aleshores els possibles nivells de gris (0-255 en imatges en grayscale) es col·loquen a l'eix 0Y, i el número de píxels de cada nivell es representa a l'eix 0X. Per aquest segon també es fan representacions amb el percentatge de píxels d'un determinat nivell de gris respecte al total de píxels de la imatge. Per imatges RGB es representen els tres canals vermell, verd i groc per separat.

[1] Aquestes imatges han sigut extretes de l'article de L. M. Quintana et al. [24].

[2] Aquestes imatges han sigut extretes de l'article de L. M. Quintana et al. [24].

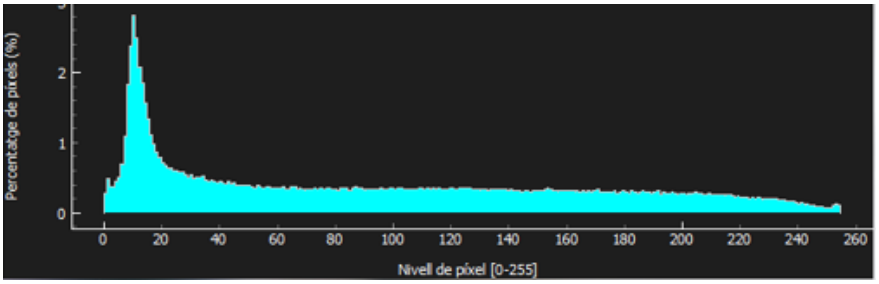

Figura 4: Exemple d'histograma d'una imatge on a l'eix 0Y es representen els nivells de píxel per a una imatge en escala de griso i a l'eix 0X el percentatge de píxels per cada nivell respecte al total de píxels de la imatge.

3.3.1.4. Overfitting i segmentació

Segons el principi de la navalla de Occam, o el principi de parsimònia, tot model i procediment ha de contenir lo exclusivament necessari per modelar, res més. És a dir, si per exemple un model de regressió necessita només dos predictors per explicar *y,* mai s'haurien d'utilitzar més de dos. L'overfitting és doncs l'ús de models o procediments que no segueixen aquesta norma, incloent més termes o plantejaments més complicats de lo necessari [26]. En el camp de la segmentació, un overfitting equival a una sobre-segmentació, i.e. a dividir regions que en realitat haurien de formar una de sola. Vegeu l'exemple a continuació:

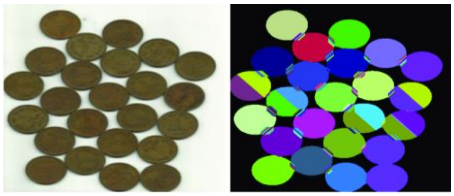

Figura 5: exemple de sobre-segmentació. A l'esquerra la imatge original i a la dreta a imatge segmentada on s'aprecien més regions de les que són necessàries.[1]

3.3.1.5. Algorismes de segmentació

Els algorismes utilitzats per anàlisis estadístics es poden separar en dues categories principals: els algorismes supervisats i els no supervisats. A continuació s'explica en detall el significat de cadascun:

[1] Aquestes imatges han sigut extretes de l'article d'A. Kornilov et al. **Fuente especificada no válida.**

- **Algorismes supervisats**

Aquests són els que necessiten de la intervenció humana per a funcionar. Les persones han d'interactuar mitjançant el control de la entrada i la sortida de dades i la valoració de la precisió. En aquests casos el model s'entrena amb dades d'entrada la sortida desitjada de la qual és coneguda. Normalment s'utilitza en aplicacions on hi ha un gran volum de dades històriques. En són exemples mètodes de classificació com l'anàlisi discriminant lineal (LDA), les xarxes neuronals convulocionals i l'anàlisi de components principals (PCA). Cal destacar que en el cas concret de l'aplicació que s'està creant en aquest projecte no s'han tingut en consideració segmentacions basades en algorismes supervisats ja que precisaria de l'ajuda d'un expert per dur-les a terme.

- **Algorismes no supervisats**

En aquest cas els algorismes utilitzen aprenentatge profund per processar feines complexes sense l'entrenament humà. L'objectiu és explorar les dades en busca de patrons [27]. És gràcies a l'autonomia d'aquest tipus d'algorismes pel que els fan idonis per la nostra aplicació.

3.3.2. Segmentació d'imatges

La segmentació d'imatges es defineix com el mètode de segregació d'imatges a sectors exclusius que es corresponen amb uns estàndards predeterminats [28]. Un exemple d'aplicació comú al camp de la medicina és la distinció de teixits normals i anormals pel reconeixement de tumors cerebrals.

A l'abordar el problema de la segmentació és important tenir present que l'objectiu que es persegueix en aquest projecte no és la extracció o reconeixement d'objectes, sinó la identificació de les regions presents a la imatge. Aquests dos termes solen causar confusió per la estreta relació que els uneix, no obstant això, són nocions diferents. Les regions serveixen com a punt de partida per a reconstruir objectes, acció que forma part d'una etapa posterior al processament i que requereix l'aplicació del coneixement [29], en el nostre cas d'un especialista en medicina. A partir de la detecció d'objectes sorgeixen un gran número d'aplicacions, entre les que es destaquen la recuperació, el anàlisis, la transmissió i la compressió d'imatges [30]. Tal com diu Cheng et al. [31], *L'etapa de segmentació no es preocupa de la identitat dels objectes.*

Tot i que a la literatura es poden trobar diverses formes de classificar les tècniques de segmentació, es distingeixen tres categories bàsiques: tècniques de segmentació basades en píxels, basades en vores i basades en regions [29], vegeu el següent esquema:

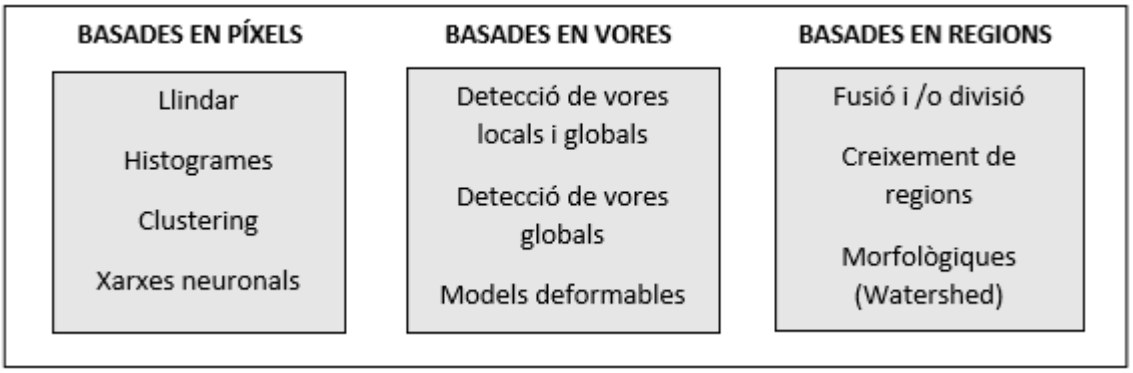

Figura 6: Classificació de les tècniques de segmentació. [29]

Cadascuna segueix un procés amb unes condicions diferents per trobar les similituds que permeten agrupar els píxels en regions. Les tècniques basades en píxels defineixen la segmentació com la tècnica que consisteix en *agrupar els píxels d'una imatge en conjunts homogenis respecte a una o varies característiques* , segons Moghaddamzadeh et al. [32]. En canvi, les basades en vores la defineixen com, segons Lucchese et al. [33], *una operació de baix nivell relativa a la partició d'una imatge [...] mitjançant la localització de les vores o fronteres.* Per últim les basades en regions, per Cheng et al. [31], *la segmentació d'imatges és el procés de dividir una imatge en diferents regions, pertanyents a diferents objectes, tals que cada regió és homogènia però si s'uneixen dos regions adjacents qualssevol deixen de ser homogènies.*

No només es diferencien aquestes tècniques per la seva metodologia sinó també pel resultat obtingut. Les tècniques de píxels agrupen els píxels semblants sense tenir en compte les seves característiques espacials, podent aparèixer aquests dispersos en diferents ubicacions a la imatge. Per una altra banda, les tècniques basades en fronteres localitzen els contorns de la regió, però no la extreuen pròpiament [29]. Les úniques tècniques que donen regions com a resultat són les basades en regions, construint-les mitjançant l'agrupació de píxels per proximitat espacial i per les seves característiques.

A continuació es descriuen i comparen aquests totes les tècniques.

3.3.2.1. Basades en píxels

Les primeres propostes de segmentació van ser tècniques de llindar, tot i que amb el temps han anat evolucionant fins a aplicar-se als histogrames de la imatge [34], [35]. En moltes ocasions s'utilitzen els pics dels histogrames per a realitzar un agrupament, com en el cas de la segmentació per barreja de gaussianes, tot i que les propostes més recents opten per xarxes neuronals.

- **Segons el llindar**

Aquestes tècniques es basen en la creació de rangs a través de diverses característiques, com poden ser el nivell de gris, el color i la textura [36]. Així, s'assumeix que els píxels el valor del qual es troben en un determinat rang pertanyen a la mateixa classe. L'inconvenient d'aquestes tècniques és que només resulten útils en imatges amb únicament dues classes: fons i objecte [37].

- **Segons histograma**

Encara que en un principi només estaven pensades per imatges en nivell de gris, poc a poc s'han anat creant versions per imatges en color, tot i que la manipulació d'un histograma en tres dimensions pot resultar massa complexa i costosa [34]. La metodologia es basa en trobar els pics més representatius al histograma per a fer una llindarització. Aleshores, a partir del valor de llindar la imatge es divideix en dues classes: valors de píxel iguals o superior al llindar i valors inferiors. La problemàtica que comporten aquestes tècniques són la forta dependència del valor del llindar [38] i la pèrdua d'alguns objectes de la imatge pel fet de tenir una mida tant petita que no ocupen suficients píxels i no es veuen reflectits al histograma [29]. Un exemple de tècnica d'obtenció de llindar automàtic és l'Otsu.

- **Agrupament (clustering)**

Aquestes tècniques parteixen d'un conjunt de vectors de característiques inicial (un per cada clúster). Posteriorment es van classificant els píxels de la imatge en funció de la seva proximitat al centroide de cada grup [39], [40]. Normalment aquests processos són iteratius, refinant la classificació a cada volta, en alguns casos fins i tot pot variar el número de clústers. Una decisió que s'ha de prendre en aquests casos és el número de clústers presents a la imatge.

Aquestes tècniques son no-supervisades i necessiten menys temps per generar els resultats. Trobem diferents subgrups dins d'aquesta categoria, com en són el clustering basat en partició, clustering basat en distribució i el clustering basat en densitat [41], en són exemples la segmentació K-Means, Gaussian Mixture Model i Quickshift, respectivament.

- **Xarxes neuronals**

Aquestes són molt utilitzades per problemes de classificació, ja que són capaces de processar paral·lelament i explorar a la vegada diferents conjunts d'hipòtesis. No obstant, l'inconvenient és que precisen de molt temps per entrenar la xarxa, el que implica que és una tècnica adequada per aplicar-la a tipus concrets d'imatges, però no quan les imatges són de naturalesa variada [42], com és el nostre cas.

3.3.2.2. Basades en contorns

Aquestes tècniques són especialment recomanables quan les regions d'ambdós costats del contorn tenen característiques molt diferents. Per desgràcia, aquesta no és una condició realista. Els pitjors resultats es fan visibles quan les regions tenen un color similar però textures diferents [29]. Segons Davis et al. [43] les tècniques de segmentació basades en vores es poden subdividir en dos grups: les locals i les globals. Davis no va tenir en compte els models deformables ja que van ser utilitzats amb posterioritat [29].

- **Tècniques locals**

Per tal de detectar i seguir el contorn de les regions, aquestes tècniques acostumen a calcular el gradient mitjançant màscares de diferencials. En aquest cas, el fet de que un píxel pertanyi a una frontera és independent de quins altres píxels ho facin. És per això que el principal problema d'aquestes tècniques és que els operadors de gradient (com poden ser Roberts, Sobel, Prewitt i Laplace) no són capaços de detectar canvis petits en les característiques dels píxels adjacents, el que provoca la detecció únicament de vores abruptes.

- **Tècniques globals**

En aquest cas, i contràriament al cas anterior, la pertinença a una frontera en un punt donat és dependent dels píxels veïns. La metodologia que segueixen les tècniques globals és la programació dinàmica o en camps de Markov.

- **Models deformables**

Aquestes tècniques defineixen polígons amb splines als costats. Això permet obtenir formes més aproximades i arrodonides. El major inconvenient resideix en que no s'adapten automàticament a les variacions de cada regió, sinó que necessiten ser ajustades inclús per cada tipus d'imatge. A més, com que tots els punts s'han d'ajustar a la vegada mitjançant la tècnica de minimització, és probable que alguns punts no s'ajustin adequadament [44].

Aquestes tècniques, tal com indica Cheng et al. [45], no poden segmentar la imatge per sí mateixes, tot i que proporcionen informació sobre els contorns que pot ser útil i utilitzada en combinació amb altres. A més, moltes de les fronteres localitzades pateixen de discontinuïtats o sobre-deteccions, el que fa que els resultats només siguin *candidats* a vores reals. És per això que aquest tipus de tècniques són les menys recomanables.

3.3.2.3. Basades en regions

Tal com Prados [29] indica, són múltiples les referències que tracten de definir el concepte de regió. Ella mateixa en fa el següent resum:

Definició 1: *Una regió Rs d'una imatge I és un conjunt de píxels semblants i connectats.*

Les tècniques basades en regions són les que proporcionen regions d'acord a aquesta definició, és a dir, els píxels que formen la regió no només tenen característiques semblants sinó que a més han d'estar connectats. Aquestes comencen des de la suposició de la existència d'un píxel o grup de píxels que són representatius de la regió que s'intenta segmentar, aquest se'ls hi anomena *llavor* [46], [47], [48]. Aleshores, el procediment continua amb la avaluació dels píxels del voltant de la llavor, intentant fusionar els que compleixin unes determinades condicions d'homogeneïtat. Quan ja no se'n troben més, es passa a la següent llavor de la següent regió i es procedeix a la mateixa metodologia.

Aquestes tècniques tendeixen a veure's menys afectades pel soroll de la imatge, a més, al incorporar tant informació de les característiques dels píxels com de les seves relacions espacials, són una de les tècniques més àmpliament utilitzades. L'inconvenient però sol ser una sobre-segmentació [29]. A continuació es comparen les diferents tècniques basades en regions.

- **Tècniques de fusió i/o divisió**

Les tècniques de fusió o divisió per separat comporten una sèrie d'inconvenients que les fa ineficients. Mentre que la tècnica de fusió parteix d'una regió petita i homogènia que es va unint successivament a les del voltant quan es compleixen les condicions de uniformitat, les de divisió comencen amb una regió gran i heterogènia que es va partint fins a quedar petites regions homogènies. El problema però, és que tendeixen a oferir com a resultat un contorn clarament quadrat, ja que es va formant la regió de forma iterativa en un número fixe de quadrats [29].

Una possible solució és la combinació d'ambdós mètodes, oferint la possibilitat de tornar a fusionar o dividir regions com més convingui. D'aquesta manera els contorns queden més realistes i suaus.

- **Tècniques morfològiques (Watershed)**

La tècnica més representativa és sens dubte la de Watershed [49], [50]. Aquesta tècnica s'ha d'aplicar sobre una imatge de gradients i analitzar-se com un mapa topogràfic, on les fronteres són representades com a muntanyes i les regions com a valls. Aquest algorisme simula una inundació on l'aigua comença a fluir a través de les llavors i es continua estenent fins a topar amb una muntanya que impedeix el seu pas, delimitant així cada regió.

Aquest mètode té l'avantatge de proporcionar vores contínues i tancades. No obstant, a vegades pot patir de sobre-segmentació.

- **Tècniques de creixement de regions**

El creixement de regions es basa en repartir llavors a diferents ubicacions de la imatge, que seran els punts de partida de cada regió. A partir d'aquests punts les regions aniran creixent condicionades a un criteri d'homogeneïtat, com poden ser la textura, la intensitat i la nitidesa [51]. És important destacar que cada llavor donarà lloc a una regió, és a dir, la imatge es dividirà en tantes regions com llavors col·loquem al principi [46], [48], [52].

Els avantatges d'aquesta tècnica són que ofereix regions amb contorns molt semblants als reals, no afavoreix regions d'una mida específica i que permeten incorporar fàcilment informació sobre la topologia de la imatge [53]. Per una altra banda, l'inconvenient és la rellevància de la primer fase de posicionament de les llavors, sent inclús dependent del ordre en que es processin les regions [29].

3.3.3. Estadística

La segmentació en que s'han focalitzat principalment els esforços d'aquest projecte s'ha realitzat mitjançant un Model de Barreja de Gaussianes (*Gaussian Mixture Model, GMM*). Els models de barreja són formats per una combinació no especificada de múltiples funcions de probabilitat. S'utilitza un procediment estadístic o un algorisme d'aprenentatge per estimar els paràmetres de les distribucions de probabilitats que s'ajusten millor a la densitat d'un conjunt de dades d'entrenament determinat [54]. És per aquest motiu pel que en aquest capítol s'explicaran els conceptes estadístics que juguen un paper clau en l'enteniment de la metodologia del model i el discerniment de les complicacions que comporta.

En concret, en la secció *2.3.3.1 Models de distribució de probabilitat,* es tractaran els conceptes de funcions de probabilitat, model de distribució de probabilitats, estimació de densitat, model de variables latents, estimació de màxima probabilitat, algorisme de maximització d'expectatives i el criteri d'informació Bayesià (BIC).

3.3.3.1. Models de distribució de probabilitat

Un dels objectius de la estadística és el de donar valors quantitatius a fets reals. Per fer-ho és necessari construir un model d'aquesta realitat que s'ajusti lo màxim possible, partint de la premissa de que lo real és sempre més complex i menys uniforme [55]. Com no és possible establir valors precisos, es fa ús de funcions de probabilitat, que es defineixen com la probabilitat de que una variable aleatòria prengui un valor en particular [56]. És per aquest motiu que els models que representen distribucions

empíriques inclouen en la seva formulació funcions de probabilitat, anomenant-se doncs models de distribució de probabilitats.

El fet de seleccionar el model de distribució de probabilitats i els paràmetres que millor expliquen la distribució conjunta de probabilitats de dades observades s'anomena estimació de la densitat. En el nostre cas, les dades aleatòries son els píxels d'una imatge amb les seves característiques de nivell, textura, etc. Al trobar un model que defineixi el comportament de cada grup de píxels es poden agrupar aquests segons les seves similituds. Aquests grups són als que anomenarem regions de la imatge.

Així, un cop seleccionat un model per barreja de Gaussianes, cal trobar els paràmetres que s'ajusten més a cada set de dades (a cada imatge), és a dir, necessitarem un mètode que estimi per a cada imatge els paràmetres del model més adients. En primera instància s'ha d'encertar el número de distribucions gaussianes que formen el nostre model (número de components del model), i després, les característiques de cada distribució gaussiana, que es corresponen amb la mitjana, la covariància i el pes de cadascuna.

- **Número de components del model**

Per a identificar el número de distribucions gaussianes que formen el model s'ha escollit el criteri d'informació Bayesià (BIC). Segons els desenvolupadors de Scikit-Learn [57], el criteri BIC ho fa de forma més eficient que l'AIC, ja que el primer penalitza molt més la complexitat del model, és a dir, evita el overfitting [58]. En teoria, el criteri d'informació Bayesià recupera el veritable nombre de components només en el cas que hi hagi molta informació disponible i suposant que la col·lecció de dades sigui independent i estigui idènticament distribuïda per distribucions Gaussianes [57].

- **Paràmetres de cada component del model**

Per a l'estimació dels paràmetres de cada distribució gaussiana, la funció utilitzada per a la segmentació per barreja de gaussianes (*GaussianMixture* de *Scikit Learn*) implementa l'algorisme de maximització d'expectatives [57]. Aquest algorisme segueix un mètode iteratiu segons una estimació de probabilitat màxima, on el model en qüestió depèn de variables latents no obtingudes [59]. Per aquest motiu, es donaran a conèixer en primer lloc els conceptes d'estimació de probabilitat màxima i model de variables latents per a procedir a la explicació del algorisme de maximització d'expectatives, sempre des del punt de vista d'un model de barreja de gaussianes.

- Models de variables latents

  Els models de variables latents es caracteritzen per representar conjunts de dades on només es poden observar algunes de les variables rellevants. Les altres, tot i que influeixen en el conjunt de dades, romanen ocultes. Més generalment, a aquestes variables que no es veuen o que s'oculten s'anomenen variables latents.

  Un model de variable latent fa suposar que una observació x és causada per alguna variable latent subjacent, una variable que no es pot observar directament, però que pot inferir a partir de variables i paràmetres observats. En el nostre cas, el número de distribucions gaussianes del model i els seus respectius paràmetres són les equivalents a les *variables latents* [60].

- Estimació de probabilitat màxima

  En estadística, l'estimació de probabilitat màxima o de màxima versemblança és un mètode d'estimar els paràmetres d'una distribució de probabilitats mitjançant la maximització d'una funció de probabilitat, de manera que sota el model estadístic assumit les dades observades són més probables. El punt de l'espai de paràmetres que maximitza la funció de probabilitat s'anomena estimació de probabilitat màxima. La lògica de la màxima probabilitat és intuïtiva i flexible, i per tant, el mètode s'ha convertit en un mitjà dominant d'inferència estadística [61].

- Algorisme de maximització d'expectatives

  La principal dificultat per aprendre models de barreja gaussiana a partir de dades sense etiquetar és que normalment es desconeix quins punts provenen de cada component latent.

  L'algorisme d'esperança-maximització (EM) és un mètode iteratiu amb l'objectiu de trobar els estimadors de màxima probabilitat dels paràmetres d'un model de distribució de probabilitats, on aquest depèn de variables latents no obtingudes [62], és a dir, d'un model de variables latents.

  El mètode primer assumeix components aleatoris (tant pot ser centrats aleatòriament en punts de dades, apresos de K-Means, o fins i tot només distribuïts normalment al voltant de l'origen). Aleshores, per a cada punt, calcula la probabilitat de ser generat per cada

component del model – per cada gaussiana-. Seguidament, els paràmetres són ajustats amb la finalitat de maximitzar la probabilitat de que les dades es corresponguin a aquestes assignacions. Al reproduir el procediment repetidament, es confereix a un òptim local [57].

4. Estat de l'art

La metodologia adoptada pel processament d'imatges biomèdiques no pot esdevenir la mateixa que per altres imatges d'escenes naturals. Les primeres presenten una problemàtica específica a causa de la necessària precisió i exactitud pròpia del diagnòstic mèdic. A més, no totes les imatges mèdiques comparteixen les mateixes característiques, sinó que estan sotmeses a variacions segons la tècnica amb la que s'han generat. En són dificultats comuns el soroll, el poc contrast i les vores mal definides o borroses [2].

És per aquest motiu pel que es necessita un certificat especial a l'hora d'usar màquines d'adquisició d'imatges mèdiques i mòduls de programari associats, sotmetent-los a rigorosos controls. En el cas d'Estats Units aquest certificat l'atorga a DFA (*Food and Drug Administration*) i a Europa la *European Medcines Agency*.

L'objectiu d'aquest capítol és fer un resum de la història del software de processat d'imatges i contemplar els reptes que presenta així com les seves limitacions.

4.1. Antecedents

A diferència del processat d'imatges general que data dels anys 70, no va ser fins als 80 que el processat d'imatges mèdiques va tenir els seus orígens. La Ressonància Magnètica Funcional va impulsar l'interès pel reconeixement mèdic a partir d'imatges al 1985. Un dels fets més importants es produeixen al 1991, a Estats Units, mitjançant un projecte amb l'objectiu de recopilar minuciosament totes les dades possibles sobre el cos humà, l'anomenat *Visible Human Project Male and Female* (VHP) de la *National Library of Medicine* (NM). Més tard, al 1999, es va crear un software gratuït per visualitzar, segmentar i registrar imatges en 3D per analitzar la informació del VHP, ja que aquesta començava a ser enorme i era necessari disposar d'eines per manipular i reconstruir la informació, a més de visualitzar-la i tractar-la per ajudar al àmbit mèdic. Va ser gràcies a aquest software que va sorgir la comunitat ITK (insight Toolkit), que disposa d'una llibreria de software lliure amb una àmplia funcionalitat que cobreix totes aquestes necessitats. Alguns dels organismes més rellevants en fan ús avui en dia, en són exemples la Clínica Mayo, la universitat de Harvard, Utah, Columbia i el Imperial College de Londres. Finalment al 1993, es va definir un protocol concret d'imatge, l'estàndard DICOM (*Digital Imaging and Communication in Medicine*) [2]. Segons A. Beraciartua [2] es defineix com *l'estàndard reconegut mundialment per a l'intercanvi d'imatges mèdiques. [...] dissenyat per al seu maneig, emmagatzematge, impressió i transmissió. Inclou la definició d'un format de fitxer i d'un protocol de comunicació de xarxa.*

No obstant, el software obert que avui en dia es més conegut per a la interacció entre usuari i imatge és l'ImageJ. Aquest ofereix la possibilitat d'aplicar filtres, processats, segmentacions i d'obtenir paràmetres. La dificultat que presenta aquest software és la complexitat i tecnicismes que el fan únicament accessible a professionals especialitzats en el processat d'imatges [1]. Així, aquest projecte planteja una interfície d'usuari amb habilitats similars a aquesta però amb la diferència d'uns resultats més intuïtius i entenedors que apropin les tecnologies als professionals mèdics.

4.2. Limitacions i reptes

Es fa front a un triple repte a l'hora de processar una imatge mèdica; en primer lloc s'ha de realitzar una segmentació precisa, aquesta segmentació ha de sorgir d'un procés automàtic i aquest procés s'ha d'executar el més ràpid possible.

La complexitat del problema resideix en aconseguir una segmentació que al ser comparada amb una segmentació de referència feta manualment per un expert les regions trobades coincideixin, evitant fugues i sobre-segmentacions. Per desgràcia, les imatges mèdiques tendeixen a presentar nivells de gris molt semblants en zones diferents i adjacents, vores difuses i baix contrast, el que fa complicada la existència d'un software que no precisi d'ajustament dels paràmetres per diferents imatges.

Una altra de les grans limitacions és la dificultat per contrastar la versemblança dels resultats. No existeixen dades de proba ni validació adequades amb les que comparar els mètodes per determinar quin és més eficient, com podria ser una base de dades d'imatges mèdiques correctament processades per cada òrgan obtingudes mitjançant diferents procediments per a d'adquisició. Ara per ara, cada estudi ha de fer ús dels mitjans que disposi per a analitzar els resultats.

Un altre repte és el treball mà a mà dels especialistes a l'àmbit de la medicina, les màquines d'adquisició d'imatges i els desenvolupadors de software [2].

5. Disseny

De la mateixa manera que a la primera part del projecte, s'ha estudiat quines funcions poden complementar el programa a desig de l'usuari i la millor manera de disposar-les. El nostre principal objectiu ha sigut crear una aplicació que resultés entenedora i intuïtiva per a qualsevol tipus d'usuari, ja que l'aplicació va destinada a persones que probablement no tenen estudis d'enginyeria.

Així, durant aquest apartat s'explicarà els estudis que s'han dut a terme i el motiu de les decisions i canvis que ha sofert l'aplicació durant aquest segon període.

L'apartat es divideix en 3 parts; públic objectiu, disseny gràfic i estructura de classes de l'aplicació.

5.1. Usuari objectiu

Abans de començar a dissenyar l'actualització de l'aplicació cal tenir en compte les característiques del usuari potencial.

Per a la creació del disseny es va parar especial atenció al tipus de client al qual dirigim el nostre producte. Tant les necessites i exigències del mateix com les seves limitacions són importants a l'hora de dissenyar les funcionalitats òptimes i l'aspecte que haurà de tenir l'aplicació.

Es va concloure que el prototip d'usuari és un professional de la medicina, que tot i poder tenir destresa amb la tecnologia, probablement no pugui desenvolupar-se satisfactòriament a l'àmbit informàtic. Podem caracteritzar el nostre client com una persona professional i d'alt nivell i exigències intel·lectuals però sense experiència ni coneixements a nivell de processat d'imatge.

Les característiques que hauria de tenir l'aplicació doncs són les següents:

- Al·lusions a l'àmbit de la medicina per establir una relació amb els usuaris.
- Aparença formal i tècnica que satisfaci les seves exigències de professionalitat.
- Alta varietat en les funcions de processat per mostrar una aplicació completa i exhaustiva.
- Capacitat d'emmagatzemament de dades i figures extretes per a la seva futura utilització.
- Múltiples recursos d'ajuda al usuari en relació a la funcionalitat del programa i als conceptes tractats en el mateix
- Funcionament intuïtiu:
    - Títols conseqüents i clarament visibles
    - Explicacions exhaustives de les funcionalitats mitjançant ToolTips
    - Llenguatge entenedor
    - Utilització d'elements visuals (icones, colors etc.)
    - Disponibilitat de manual d'usuari
- Capacitat d'arribada a usuaris internacionals.

5.2. Casos d'aplicació

En enginyeria de programari, un cas d'ús és una tècnica per a la captura de requisits potencials d'un nou sistema o una actualització de programari. Cada cas d'ús proporciona un o més escenaris que indiquen com hauria d'interactuar el sistema amb l'usuari o amb un altre sistema per aconseguir un objectiu específic. Normalment, en els casos d'usos s'evita la utilització d'argots tècnics, preferint en el seu lloc un llenguatge més proper a l'usuari final. [63]

En el nostre cas s'ha reproduït un diagrama de casos d'ús on es mostra les accions que l'usuari pugui desitjar i com es relacionen entre sí.

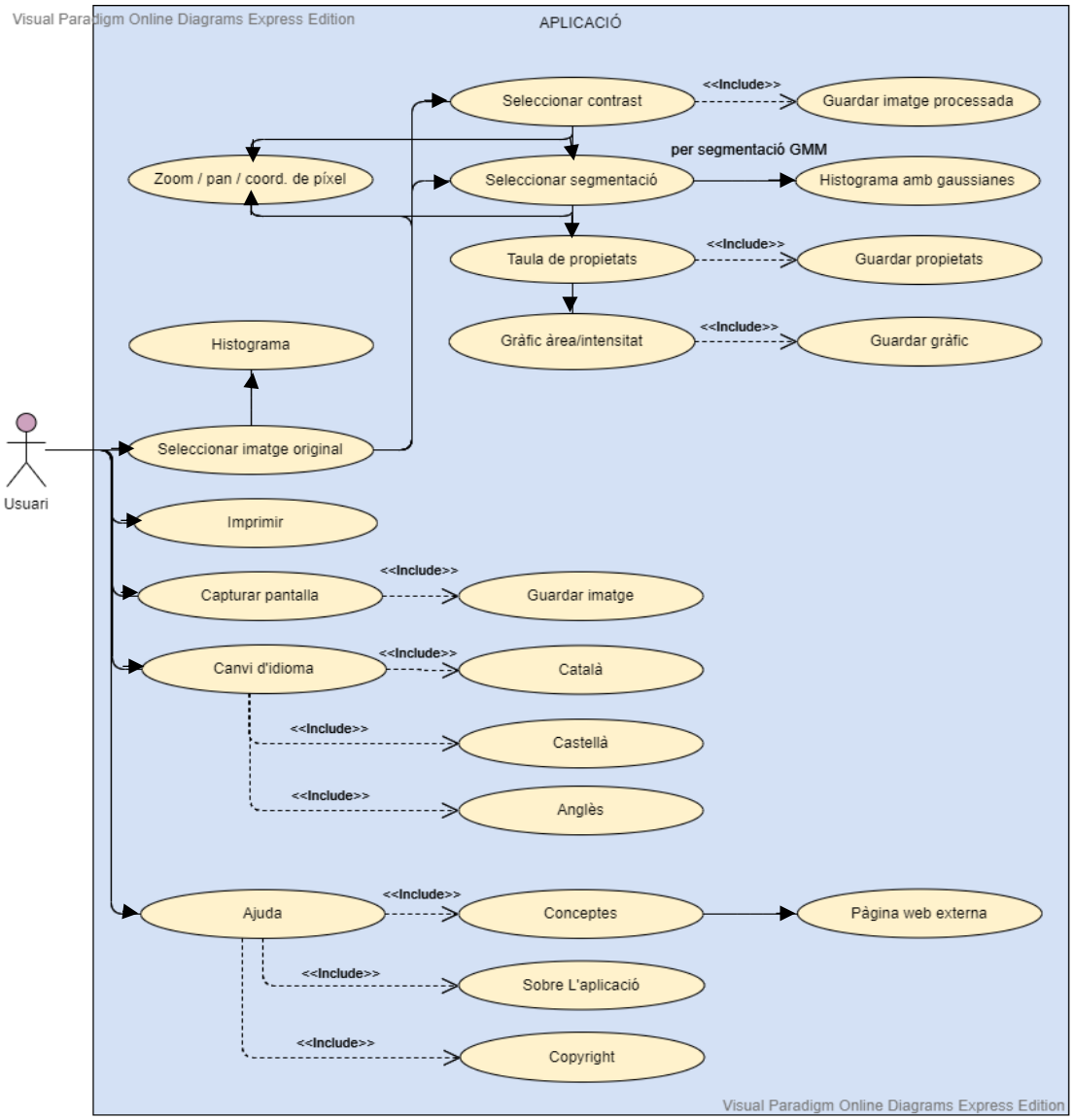

Figura 7: Diagrama UML dels casos d'ús del programa.

5.3. Disseny gràfic

5.3.1. Pantalla principal

Un dels motius que va motivar aquesta segona etapa de la interfície d'usuari va ser poder canviar l'estil de la pantalla principal. Un requisit indispensable per a que qualsevol invenció o actualització tingui èxit és que sigui atractiu, és per això que es va posar especial atenció al disseny de l'aplicació.

A continuació es mostra la disposició i aspecte de la interfície anterior i la nova, tot especificant els canvis que s'han produït.

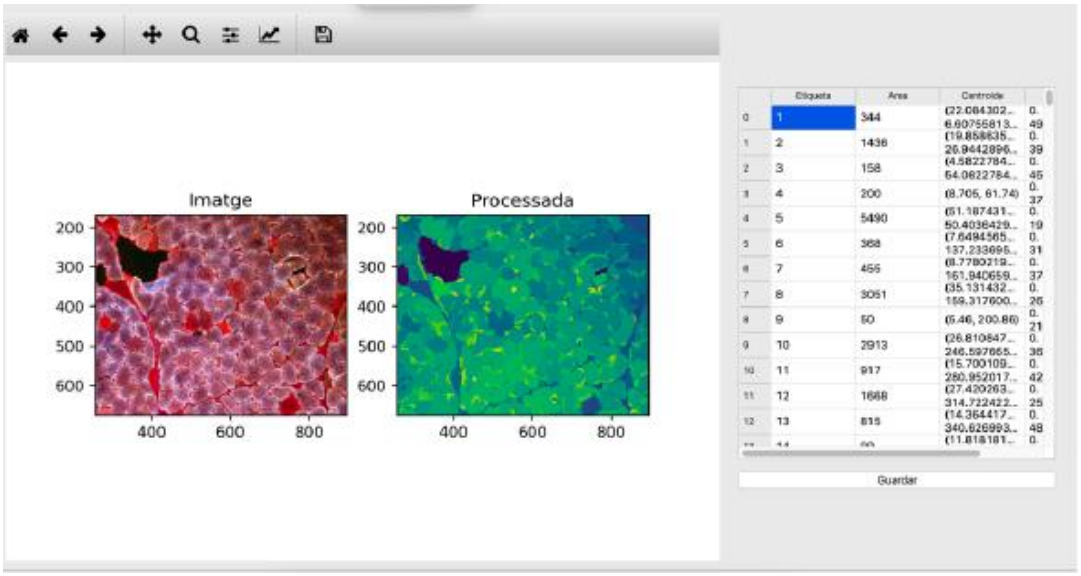

Figura 8: Disseny gràfic de la pantalla principal de l'aplicació durant la primera etapa del projecte [1].

Figura 9: Disseny gràfic de la pantalla principal de l'aplicació durant la segona etapa del projecte.

5.3.1.1. Fons de l'aplicació, colors i icones

Com es pot observar, el canvi principal ha sigut la imatge de fons i la disposició i disseny dels widgets. Es va dur a terme un estudi intensiu sobre el tipus de colors i fons que s'havien d'utilitzar a l'aplicació per expressar el missatge idoni al client.

Així, es va optar per una imatge de fons que presenti una certa relació amb l'àmbit de la medicina. Cal destacar que tot i estar ambientat a un entorn clínic, també és perfectament utilitzable en altres àmbits com el de la robòtica o el de control de qualitat.

En relació amb els colors, s'han utilitzat tonalitats grises fosques als títols i al fons de l'histograma - que suposen la majoria de la pantalla - per crear una sensació d'elegància i professionalitat, mentre que el color vermell s'ha fet servir per eines distintives de l'aplicació, com ha sigut la Toolbox. Finalment es va optar per colors verdosos per a la taula de característiques, en part per a diferenciar les eines de display i les d'obtenció de dades, i en part per crear una sensació de familiaritat i senzillesa. Per aquest mateix motiu es van afegir icones a totes les funcions que ho permetessin.

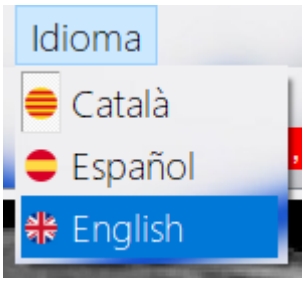

Figura 10: Exemple d'icones per fer més visuals les funcions del programa.

Mentre que originalment trobàvem una pantalla en blanc amb les dues imatges situades al centre, la taula a la dreta d'aquestes i la Toolbox comú a la part superior, a la nova actualització es va haver de fer espai per representar l'histograma, fent que es redistribuïssin la resta de widgets.

5.3.1.2. Toolbox

Una toolbox es descriu com el set de funcions o eines de software accesibles desde un únic menú. Amb motiu de la necessitat de les eines que oferia la Toolbox pròpia de Matplotlib durant el primer període, es va desenvolupar una que contingués les mateixes possibilitats. Les imatges es disposen a la part superior, amb una Toolbox pròpia per a cadascuna. Aquesta consta, al igual que durant el primer període, d'eines de zoom, arrossegament de la imatge i visualització de les coordenades dels píxels, com es pot veure a la Figura 11.

Les eines de la Toolbox estan lligades a accions de ratolí; per fer *zoom in* l'usuari ha de lliscar la roda del mateix cap amunt, per fer *zoom out* l'ha de lliscar cap avall. La funció d'arrossegar la imatge i veure les coordenades de píxel van connectades a un mateix botó, ja que són incompatibles. Al clicar sobre el botó vermell es canvia d'una funció a l'altra, i per diferenciar-les cal fixar-se en la forma del cursor; una fletxa per les coordenades i una mà per arrossegar la imatge.

Figura 11: Aparença de la Toolbox actualitzada a la segona etapa del projecte, marcada en vermell.

Figura 12: Cursor en les seves dues possibles formes. A l'esquerra amb la opció d'arrossegar habilitada i a la dreta les coordenades de píxel.

5.3.1.3. Taula de característiques

Els canvis que la taula ha sofert han sigut el canvi de posició, el canvi de color, l'afegiment d'un títol i especificacions d'unitats per cada columna. A més, a la columna de intensitat se li ha multiplicat els resultats per 100 per a que siguin més visuals, tot especificant aquesta modificació. Es poden apreciar aquests canvis a la Figura 13.

Un altre fet important sobre la taula és que durant la primera etapa del projecte, a l'hora de carregar les característiques d'una segmentació es repetia abans el procés de segmentació de la imatge. En altres paraules, tot i haver dut a terme, per exemple, la segmentació Otsu, si l'usuari accionava la visualització de les propietats de la imatge amb segmentació Otsu, aquesta es tornava a repetir, augmentant enormement el temps d'execució. Això no representava un problema pels primers tipus de segmentació, que es podrien qualificar de més directes o senzills, però un cop es van implementar segmentacions més complexes aquesta repetició comportava una pèrdua d'eficiència.

En conseqüència, durant la segona etapa del projecte es va reduir aquest codi innecessari creant una taula a partir de la última segmentació realitzada a la imatge, fent també que a la barra de menú només aparegués una opció per a mostrar la taula de característiques, com es pot apreciar a la Figura 14.

Per evitar qualsevol confusió, es va afegir un ToolTip al botó de mostrar la taula de característiques especificant que es faria sobre la última segmentació realitzada, i una apartat específic en la secció d'ajuda a l'usuari on s'explica amb detall.

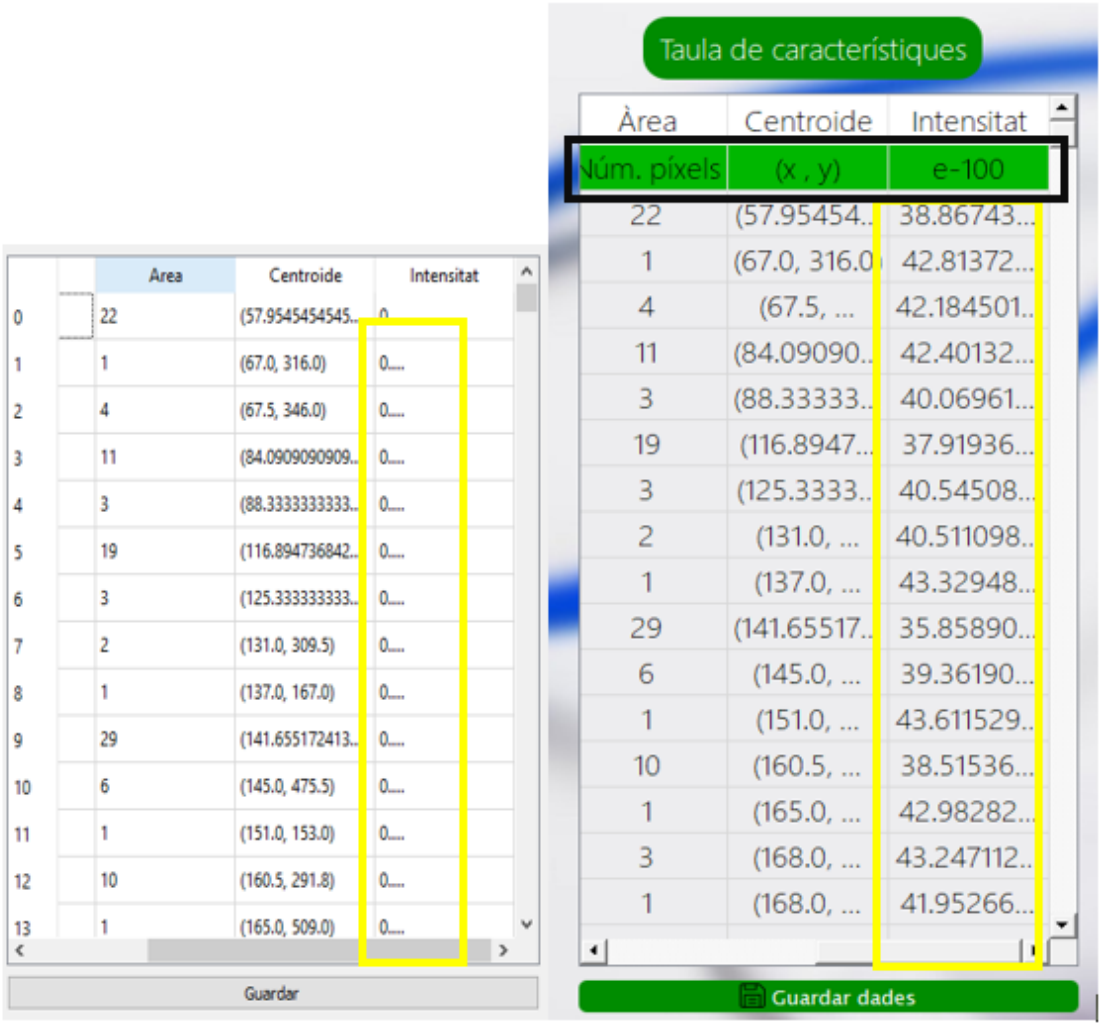

Figura 13: Taula durant el primer i el segon període del projecte, respectivament. En groc es pot apreciar el canvi a la intensitat, mentre que en negre es senyalen les especificacions de cada columna.

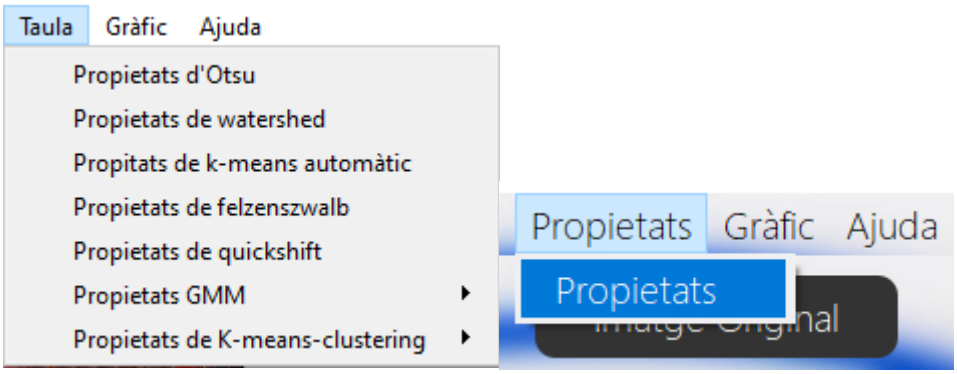

Figura 14: Comparació d'aparença de la opció de taula de característiques de la barra de menú entre el primer i segon període del projecte. S'aprecia la simplicitat de l'actualització al no haver d'especificar quina segmentació s'ha dut a terme.

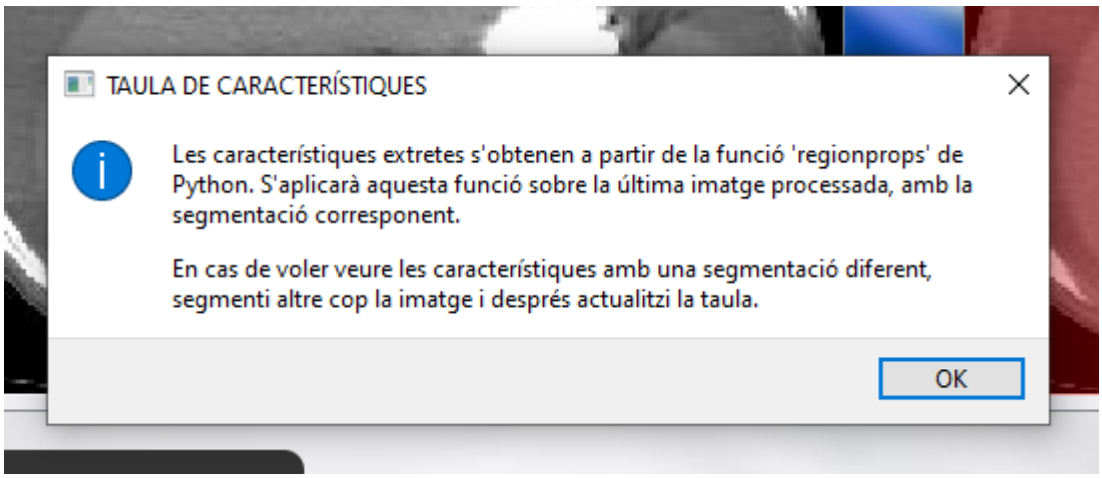

Figura 15: Exemple de finestra emergent amb ajuda per a l'usuari. En aquest cas s'especifica la metodologia seguida per a la representació de la taula de propietats i s'explica com procedir per a canviar-la.

5.3.2. Gràfic àrea / intensitat

Com s'ha comentat anteriorment, per la funció del gràfic Àrea intensitat s'ha mantingut l'eina Matplotlib, el que ha suposat una variació menor. Bàsicament es va canviar l'aspecte del plot i es va afegir un títol per fer-lo més atractiu.

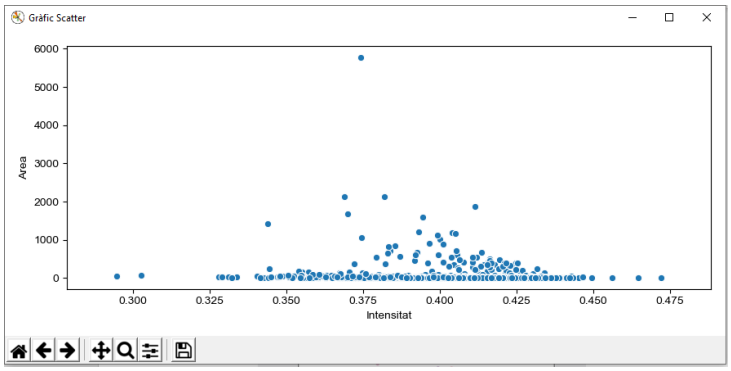

Figura 16: Disseny gràfic de la representació de característiques de la imatge durant el primer període.

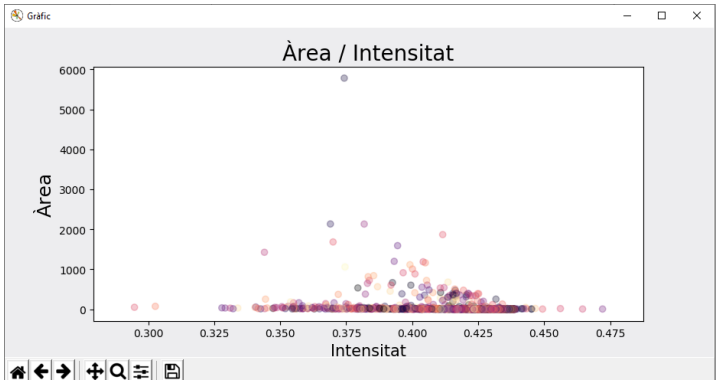

Figura 17: Disseny gràfic de la representació de característiques durant el segon període.

5.4. Estructura de classes

En aquest apartat s'especifiquen les relacions entre dades durant el primer i el segon període del projecte, tot indicant les similituds i referències entre classes i les funcions i mètodes nous s'han implementat.

5.4.1. Estructura de classes durant el primer període

De les 9 classes necessàries al primer període se'n diferencien tres tipus segons la funció que desenvolupaven; classes on es gestionaven les dades, és a dir Scene, Segmentada i RegionProps, classes de disseny de pantalla, únicament la classe Projecte, i classes de suport que connecten les dues anteriors, on trobàvem CentralWidget, PandasModel, PlotCanvas, PlotScat i TableWidget.

5.4.1.1. Classes de gestió de dades

Aquestes classes s'encarregaven de les funcions de processament, segmentació i extracció de dades. Seguidament, a la Figura 18, s'aprecia el tipus de relació que tenien entre elles.

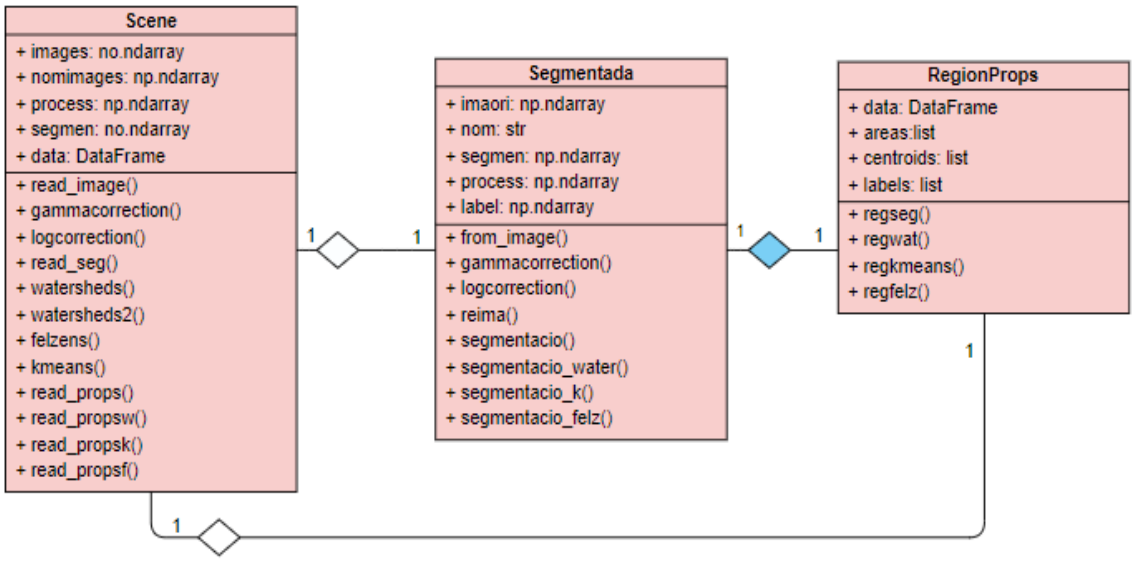

Figura 18: UML de la estructura de classes de gestió de dades durant la primera etapa del programa. Dissenyat amb draw.io. [1]

La classe Scena estava relacionada amb la classe Segmentada per agregació, el que implica que aquesta formava part de la classe Scena, però la destrucció de la classe contenidora no implicava la desaparició de la classe continguda. A la vegada, RegionProps es relacionava amb la classe Scena per agregació, i amb la classe Segmentada per composició, és a dir, si la classe Segmentada desapareixia, RegionProps ho feia a la vegada. [1]

- **Classe Segmentada**

La classe Segmentada tenia com a atributs la imatge original, el nom de la imatge, la imatge processada, la imatge segmentada i la imatge etiquetada.

Els mètodes propis d'aquesta classe tenien la funció de llegir la imatge original que havia sigut seleccionada per l'usuari i el seu nom, tot guardant-los en atributs d'aquesta. També disposava de mètodes que aplicaven el processament pertinent, és a dir, el que l'usuari demanava. Depenent del tipus de processat al que es sotmetia la imatge (contrast o segmentació) el resultat era guardat en un o altre atribut, per tenir accés a la informació amb posterioritat. Així, si la imatge havia sofert un canvi per contrast i posteriorment es volia segmentar, el programa segmentava directament la imatge amb contrast i no la original.

- **Classe RegionProps**

Els atribut eren la taula de propietats anomenada data del tipus DataFrame, llistes de les àrees, centroide i etiquetes de cada segment.

Els mètodes s'encarregaven de la lectura de propietats de la imatge un cop feta la segmentació pertinent i guardava les dades a un altre atribut.

- **Classe Scene**

Scene contenia la imatge original, la imatge processada, la imatge segmentada i un DataFrame, estructura de dades típica de Pandas que guarda taules amb noms a cada columna, amb les propietats pertinents dels segments de la imatge.

La classe Scene era el nexe entre les dades i la interfície gràfica, el que significa que cridava tots els mètodes de les classes Segmentada i RegionProps i s'emmagatzemaven les dades per poder ser utilitzades per altres classes posteriorment.

Per tant, per afegir noves tècniques de segmentació de la imatge, s'havien de dissenyar i afegir dins la classe Segmentada com a nous mètodes. Després cridar els resultats a la classe RegionProps per extreure els atributs i després incloure la imatge segmentada i la taula amb les propietats de cada segment en la classe Scena. [1]

Tot aquest procés era llarg i tediós, el que va impulsar una reducció de codi i classes durant el segon període del projecte, per tal de fer-lo més senzill i entenedor i estalviar temps d'execució.

5.4.1.2. Classes de disseny de pantalla

Gràcies a la classe Projecte es creaven els menús, widgets i botons. Es definia el disseny de la pantalla principal amb la distribució de widgets i es connectaven els botons amb els mètodes de les classes de gestió de dades. Per fer això es cridava a la classe Scene, ja que aquesta contenia totes les dades.

5.4.1.3. Classes de suport i relació software – interfície d'usuari

Per acabar, eren necessàries classes que establissin una relació entre les representacions de dades i la interfície d'usuari, ja que les eines de taules i plots de Matplotlib no estan predeterminats per aquesta funció. Aquestes classes definien la maquetació (layout) dels plots i el disseny dels mateixos. Constava de 5 classes d'aquest tipus: CentralWidget, PlotCanvas , PandasModel, TableWidget i PlotScat.

Mentre que CentralWidget i PlotCanvas servien per mostrar les imatges que s'havien obtingut amb eines de Matplotlib, PandasModel formava la taula de propietats i TableWidget la mostrava. Finalment, la classe PlotScat disposava el gràfic Àrea/Intensitat.

Com es pot observar, la complexitat i necessitat d'un gran número de classes per a establir aquesta relació feia del procés una tècnica ineficient. Vegeu l'exemple de diagrama de flux per segmentar una imatge i mostrar-la de la Figura 19.

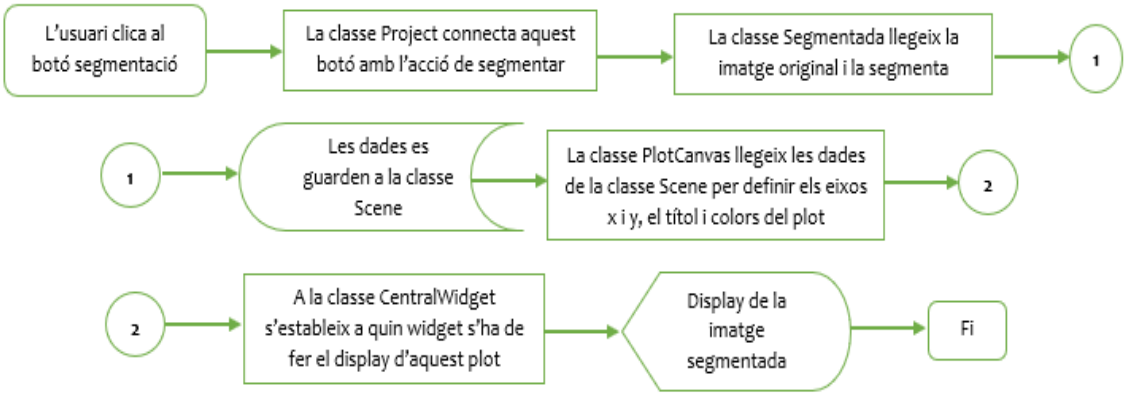

Figura 19: Diagrama de flux que mostra totes les passes necessàries per segmentar una imatge i mostrar-la a la pantalla amb l'antiga relació entre classes.

La taula de propietats de la imatge segmentada seguia un trajecte semblant, vegeu l'exemple de diagrama de flux per mostrar les propietats de la imatge segmentada en una taula:

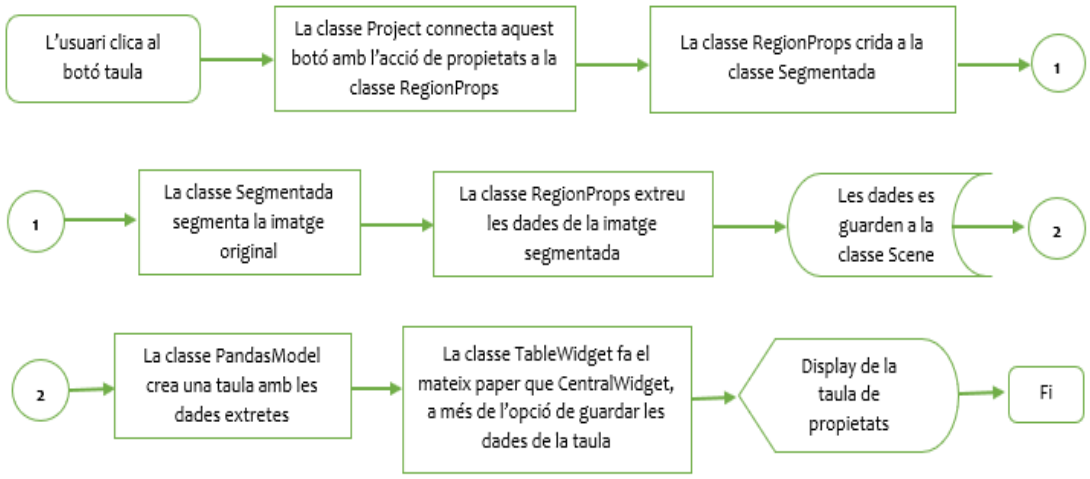

Figura 20: Diagrama de flux que mostra totes les passes necessàries per mostrar la taula de propietats d'una imatge segmentada amb l'antiga relació entre classes.

Com a conseqüència, durant el segon període del programa es va focalitzar l'atenció en reduir el codi i establir relacions software-GUI més senzilles i directes.

5.4.2. Relacions entre classes durant el segon període

Durant el segon període, i gràcies a la utilització de widgets propis de Qt, es va poder reduir el número de classes de 9 a únicament 3: Ui_TFG, Segmentada i PhotoViewer.

La classe PhotoViewer serveix únicament per crear un objecte de display d'imatges que fa possible el zoom i l'opció de veure les coordenades de píxel a la imatge quan es clica sobre ella. Per una altra banda, Segmentada és la classe que conté qualsevol mètode que impliqui interactuar amb la imatge, ja sigui processament, segmentació, histograma o taula de propietats. Per últim, la classe Ui_TFG és on es disposa tot el disseny gràfic i es fan els enllaços entre els botons i les accions. A Ui_TFG també figuren els mètodes que no interactuen directament amb la imatge, com fer captura de pantalla, canviar l'idioma i l'apartat d'ajuda.

El tipus de relació entre classes és el següent:

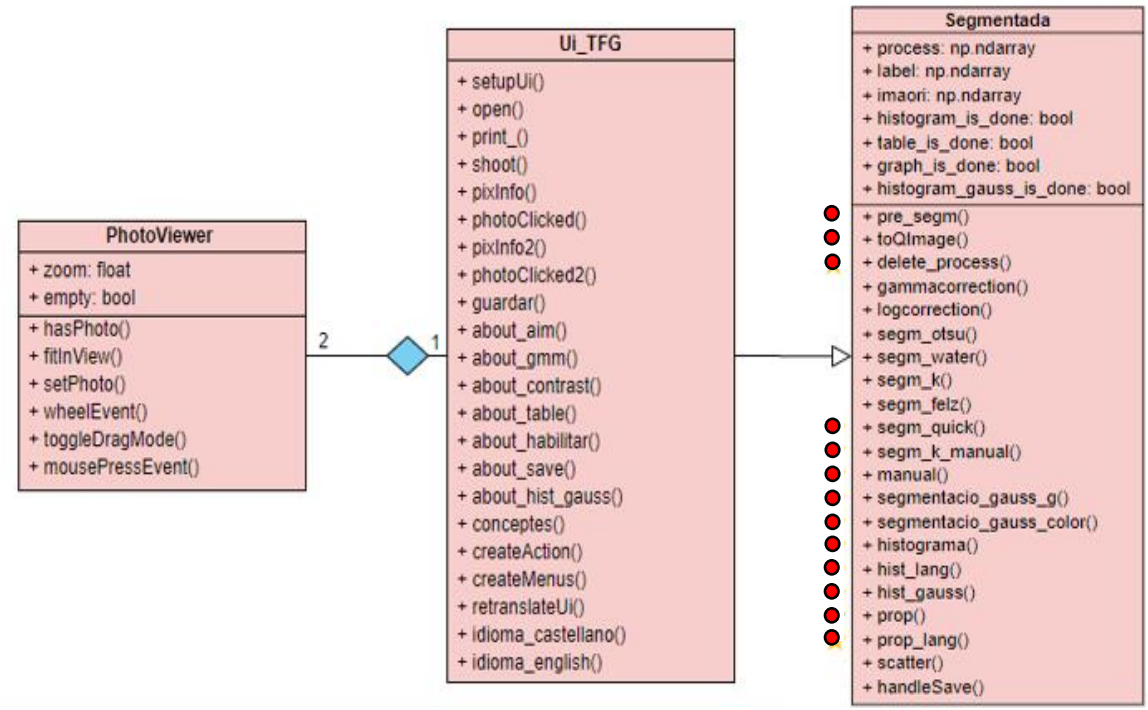

Figura 21: UML de la estructura de classes de gestió de dades durant la segona etapa del programa. Els mètodes especificats amb un punt vermell a la classe Segmentada són de nova creació.

PhotoViewer i Ui_TFG tenen una relació de composició, ja que la seva durada de vida és la mateixa. A Ui_TFG sempre li correspondran dues instàncies de la classe PhotoViewer ja que ha de mostrar tant la imatge original com la imatge processada.

Per una altra banda, Ui_TFG hereta de la classe Segmentada els seus mètodes i atributs. El que significa que la classe Segmentada és el pare i Ui_TFG el fill.

La classe Segmentada ha sigut la única que s'ha basat en el codi que hi havia principalment a l'aplicació (ja que al canviar completament el disseny de la pantalla no es podia aprofitar res de la resta de classes). S'ha especificat, mitjançant un punt vermell, els mètodes que són completament nous. També hi ha una excepció, el cas del mètode "prop()", on s' ha aprofitat la idea del que hi havia a la classe RegionProps antigament però la seva codificació és completament diferent. Igualment, cap dels mètodes que havia en un principi s'ha conservat íntegrament.

Seguidament s'explicarà detalladament la funció que duu a terme cada un dels mètodes.

5.4.2.1. Segmentada

- **pre_segm():**

Aquesta és una funció pensada per estalviar codi. Està basada en el fet que qualsevol tipus de segmentació segueix el mateix procediment d'inicialització, és per això que cada segmentació comença amb la crida d'aquesta funció.

Es comença habilitant l'acció de taula de propietats (ja que un cop feta la segmentació ja és possible extreure les característiques), després es comprova si la imatge ha sofert un canvi de contrast i de ser així estableix aquesta com a imatge a segmentar. En cas de no haver-se produït, llegeix la imatge original.

- **toQImage():**

Un altre cop és un mètode pensat per estalviar codi. Canvia el format de la imatge resultant, que de per se ens retorna un numpy array i hem de convertir a QImage per poder visualitzar-la amb eines de Qt.

- **histograma():**

Mètode que calcula i mostra l'histograma de la imatge original. El widget utilitzat permet fer ús d'una sèrie d'eines entre les que s'inclou: afegir un *grid* tant per l'eix OX com OY, invertir els eixos, habilitar o inhabilitar el ratolí, mostrar les dades de qualsevol dels dos eixos en mode logarítmic, calcular la transformada de Fourier, mostrar el màxim i la mitjana i per últim exportar les dades. Es pot escollir entre exportar el plot de l'histograma en format imatge, finestra de Matplotlib o Scalable Vector Graphics (SVG). També es pot exportar les dades en format .csv. En cas de fer-ho com a imatge es pot escollir el color de fons.

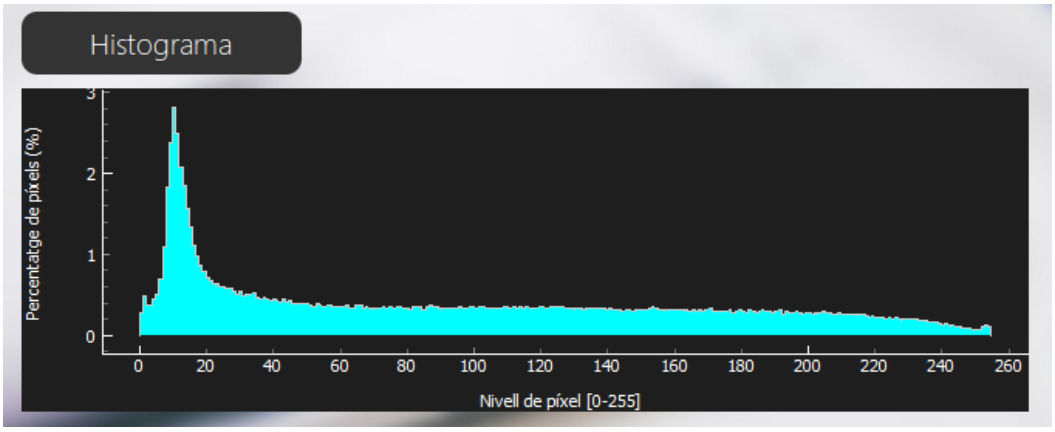

Figura 22: Exemple del resultat de la funció histograma.

- **delete_process():**

Aquest mètode va sorgir en pensar que podia donar-se el cas que un usuari hagués aplicat un canvi de contrast i després volgués segmentar la imatge sense aquest canvi. Així, aquest mètode està connectat al botó escombraria que apareix sota la imatge processada i serveix per esborrar el contrast de la mateixa.

Figura 23: Botó escombraria

Un cop s'esborra el contrast s'avisa al usuari de que s'ha produït el canvi satisfactòriament.

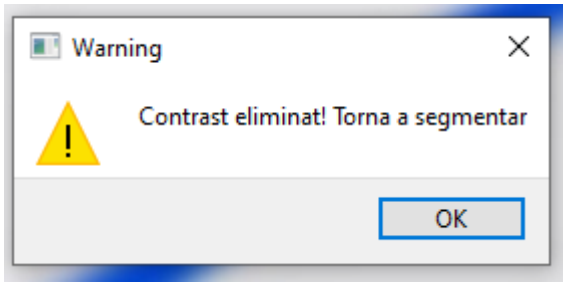

Figura 24: Finestra emergent quan s'elimina el contrast avisant al usuari de que s'ha produït el canvi satisfactòriament.

- **gammacorrection():**

Aplica un processament de contrast gamma definit en la classe Segmentada i en guarda la imatge resultant a l'atribut process. [1]

- **logcorrection():**

Aplica un processament de contrast logarítmic definit en la classe Segmentada i en guarda la imatge resultant a l'atribut process.[5]

- **segm_otsu():**

Aplica la segmentació per llindar 0tsu definida en la classe Segmentada i en guarda la imatge resultant a l'atribut segmen.[5]

- **segm_water():**

Aplica la segmentació per Watershed definida en la classe Segmentada i en guarda la imatge resultant a l'atribut segmen.[5]

- **segm_k():**

Aplica la segmentació per K-means definida en la classe Segmentada i en guarda la imatge resultant a l'atribut segmen.[5]

- **segm_felz():**

Aplica la segmentació per felzenszwalb definida en la classe Segmentada i en guarda la imatge resultant a l'atribut segmen.[5]

- **segm_quick():**

Aplica la segmentació per quickshift definida en la classe Segmentada i en guarda la imatge resultant a l'atribut segment.

[1] Definició original de S. Bernaus [1] al primer període del projecte.

- **segm_k_manual():**

Després d'haver implementat el mètode de K-Means mitjançant la funció de Python que calcula automàticament el número de clústers òptims per la imatge, es va pensar en un mètode que deixés a elecció de l'usuari aquesta decisió.
Per demanar el valor desitjat a l'usuari es crea un pop-up com el següent:

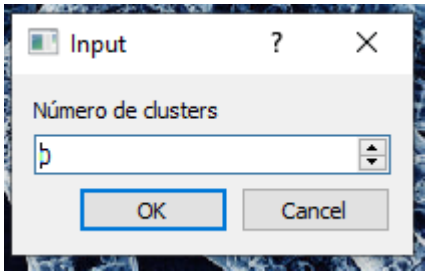

Figura 25: Finestra emergent que apareix quan l'usuari selecciona una segmentació per K-Means manual

També és important destacar que no es permet a l'usuari introduir un valor inferior a dos, major a cinquanta o un real. En aquest cas apareixeria el següent pop-up:

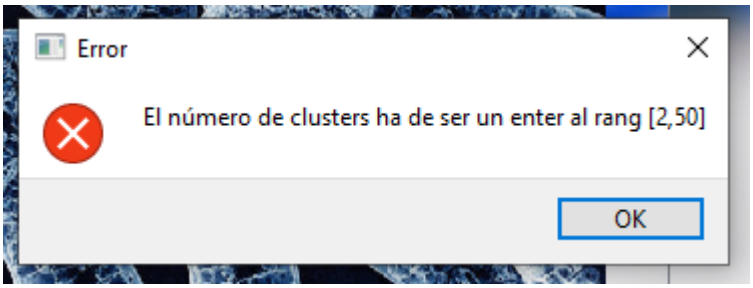

Figura 26: Missatge d'error al escollir un número de clústers fora del rang permès.

- **prop():**

Aquest mètode calcula a través de la funció *regionprops* de skimage les dades a mostrar a la taula per a cada etiqueta. Aquestes són el número de píxels que formen l'àrea, el centroide en coordenades (x,y) i la intensitat.

- **scatter():**

Aquest mètode crea una finestra nova amb el plot del gràfic àrea intensitat.

- **manual(), hist_lang(), prop_lang():**

Els mètodes que impliquen mostrar text a la pantalla estan dividits en dues parts: la part del codi on s'especifica el que dirà el text i la part que no. Aquests 3 mètodes són les meitats responsables del text corresponents als mètodes segm_k_manual(), histograma() i prop(), respectivament.

S'ha fet d'aquesta manera en el cas de voler canviar d'idioma. S'explicarà amb més detall a continuació a l'apartat **idioma_english(), idioma_castellano():**.

- **handleSave():**

Guarda les dades de la taula en format .csv en el fitxer i amb el nom que l'usuari desitgi.

- **segmentació_gauss_g(), segmentació_gauss_color(), hist_gauss():**

Aquestes tres funcions fan relació a la segmentació per barreja de gaussianes. Com es pot apreciar n'hi ha de dos tipus; per imatges grayscale i per imatges RGB. Com que es va dedicar gairebé la meitat del temps de creació de l'aplicació a desenvolupar aquesta funció s'explicarà detalladament el procés que es va dur a terme en un apartat a continuació.

Un fet remarcable és que l'histograma amb gaussianes ha d'estar només habilitat si s'ha fet una segmentació GMM-grayscale. Conseqüentment, a cada segmentació i processament s'ha d'inhabilitar la opció d'histograma amb gaussianes en cas que l'usuari l'hagi fet amb anterioritat (habilitant la opció) però ara estigui segmentant la imatge amb un altre mètode.

5.4.2.2. Photoviewer

- **hasPhoto():**

Mètode per saber si s'ha escollit imatge.

- **fitInView():**

Mètode per encaixar la imatge al widget sense modificar-la.

- **setPhoto():**

Mostrar la imatge un cop s'ha escollit.

- **wheelEvent():**

Permet fer zoom in i zoom out mitjançant la roda del ratolí.

- **toggleDragMode():**

Canviar el mode per arrossegar la imatge.

- **mousePressEvent():**

Canviar el mode per mostrar les coordenades de píxel.

5.4.2.3. Ui_TFG

- **setupUI():**

Mètode on es defineixen tots els widgets i el disseny gràfic de la GUI.

- **open():**

Mètode que utilitza un QFileDialog per accedir al fitxer del ordinador de l'usuari per escollir la imatge original a processar.

També habilita les accions d'histograma, contrast i segmentació de la imatge.

- **print_():**

Mètode que permet imprimir la imatge de forma física gràcies a QPainter.

- **shoot():**

Mètode que realitza una captura de pantalla i la guarda amb la data (dia i hora) en que s'ha pres, al mateix directori en el que l'usuari té l'aplicació guardada.

En el moment en que es guarda la imatge apareix un pop-up com el següent:

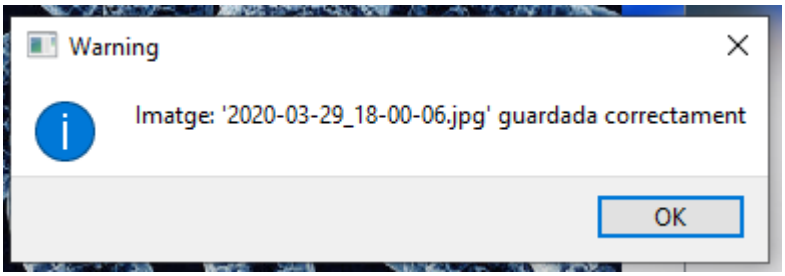

Figura 27: Warning avisant al usuari de que la imatge s'ha guardat correctament.

- **pixInfo(), pixInfo2(), photoClicked(), photoClicked2:**

Mètodes que canvien del mode arrossegar imatge a visualitzar les coordenades dels píxels al clicar gràcies a les funcions desenvolupades a la classe PhotoViewer.

- **guardar():**

Mètode que guarda la imatge processada de la mateixa manera que la captura de pantalla, essent el nom la data del moment en que es realitza i amb un warning avisant de que s'ha guardat correctament.

- **about_aim(), about_gmm(), about_contrast(), about_table(), about_habilitar(), about_save(), about_hist_gauss():**

Durant aquesta segona etapa del programa, no només s'ha focalitzat en les funcions que fan que l'aplicació quedi completa, sinó també en informar a l'usuari tant del funcionament del mateix com dels conceptes que es tracten. Finalment, s'ha afegit informació sobre el copyright.

Així, cada un d'aquests mètodes estan connectats amb el menú d'ajuda i expliquen una part de l'aplicació, entre els que trobem:

1) L'objectiu del programa

2) Com esborrar el contrast un cop aplicat per fer la segmentació.

3) De quina segmentació s'extreuen les característiques per la taula de propietats i com canviar-ho. També s'especifica el mètode que es segueix.

4) Com passar del mode arrossegar imatge a veure les coordenades, així com de quina manera es fa el zoom in i zoom out a la imatge original i processada.

5) De quina forma, amb quin nom i on es guarden tant les captures de pantalla com la imatge processada.

6) En què es basa un histograma amb gaussianes, quina informació útil aporta i en quins casos es pot utilitzar.

Vegeu l'exemple de la Figura 28 on es mostra la informació sobre segmentació i contrast.

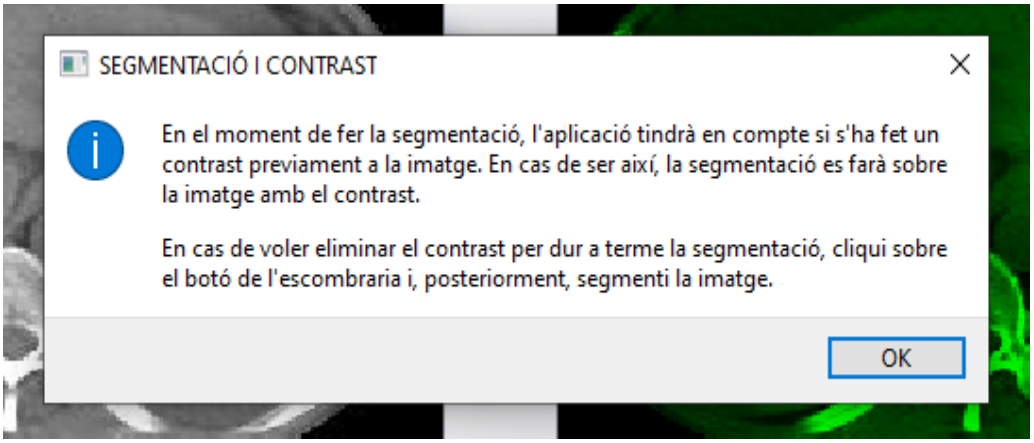

Figura 28: Exemple de finestra emergent d'ajuda per l'usuari. En aquest cas en relació al contrast.

- **conceptes:**

Per usuaris principiants en temes de processat d'imatge també s'han afegit links que redirigeixen a pàgines oficials on s'expliquen aquests conceptes. Els que s'han considerat indispensables han sigut Python, PyQt5, histograma, contrast, segmentació i GMM.

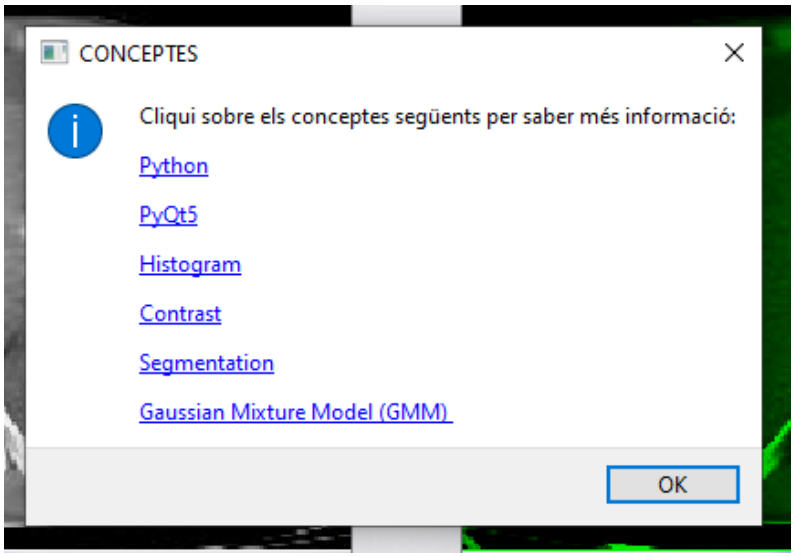

Figura 29: Finestra emergent d'ajuda sobre els conceptes que es tracten a l'aplicació.

- **createAction():**

Mètode on es connecta cada botó amb l'acció corresponent.

- **createMenus():**

Mètode on es crea la barra de menú amb tots els submenús.

- **retranslateUi():**

Mètode on es defineixen els títols de cada botó de la GUI.

- **idioma_english(), idioma_castellano():**

També es va voler ampliar l'espectre de possibles clients fent disponible l'aplicació al complet en diferents idiomes, entre els que trobem el català, el castellà i l'anglès. Aquests mètodes s'encarreguen de canviar l'idioma a anglès i castellà, respectivament. Per tornar a idioma català, el botó està connectat amb el mètode anterior (retranslateUi()).

Hem de tenir en compte el possible cas següent: un usuari obre l'aplicació en un idioma i utilitza un widget que mostra text (histograma, taula o gràfic) i , més tard, decideix canviar d'idioma. Per a que es canviï aleshores també l'idioma del text mostrat pel widget sense que l'usuari hagi de tornar a carregar la funció, hem d'actualitzar la meitat del mètode amb text.

És per això que els mètodes que mostren text en pantalla s'han separat en 2 parts; la que conté el text i la que no. I és també per aquest motiu pel que s'ha creat un atribut que especifica si s'ha accionat el mètode amb anterioritat, ja que no podem cridar a la funció si l'usuari no ho havia fet abans. Així, podem saber si l'usuari havia fet ús d'aquest tipus de widgets abans de demanar canviar d'idioma. En cas afirmatiu, es crida a la part del mètode responsable del text per a que s'actualitzi la llengua.

Vegeu el següent exemple:

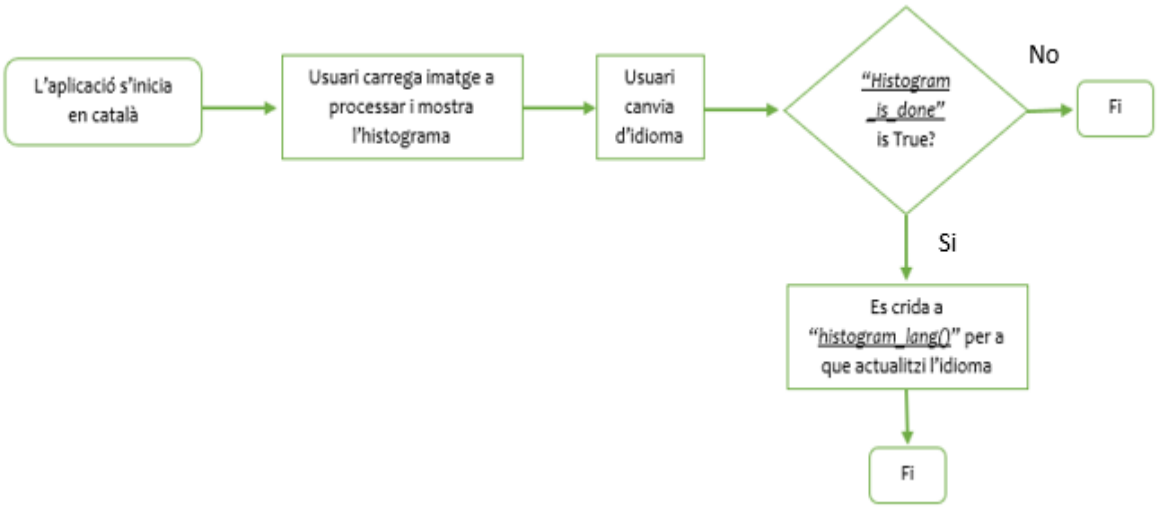

Figura 30: Diagrama de flux per explicar la raó de la divisió d'alguns mètodes en 2 i crear els atributs "is_done".

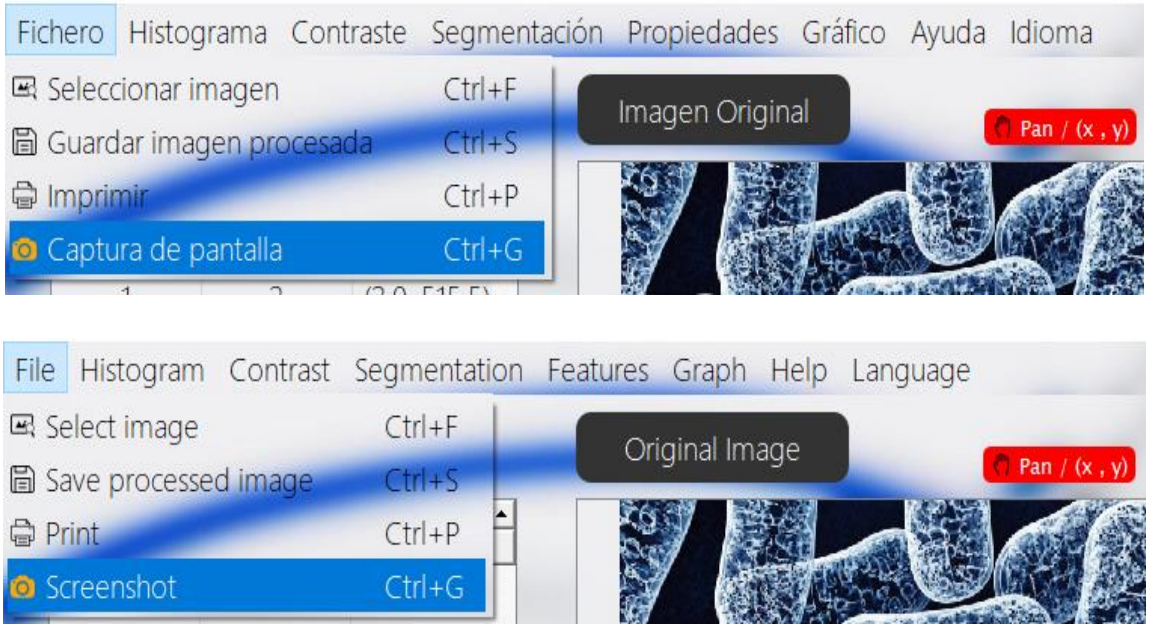

Figura 31: Exemple del canvi d'idioma tant al castellà com a l'anglès dels títols i barra de menú.

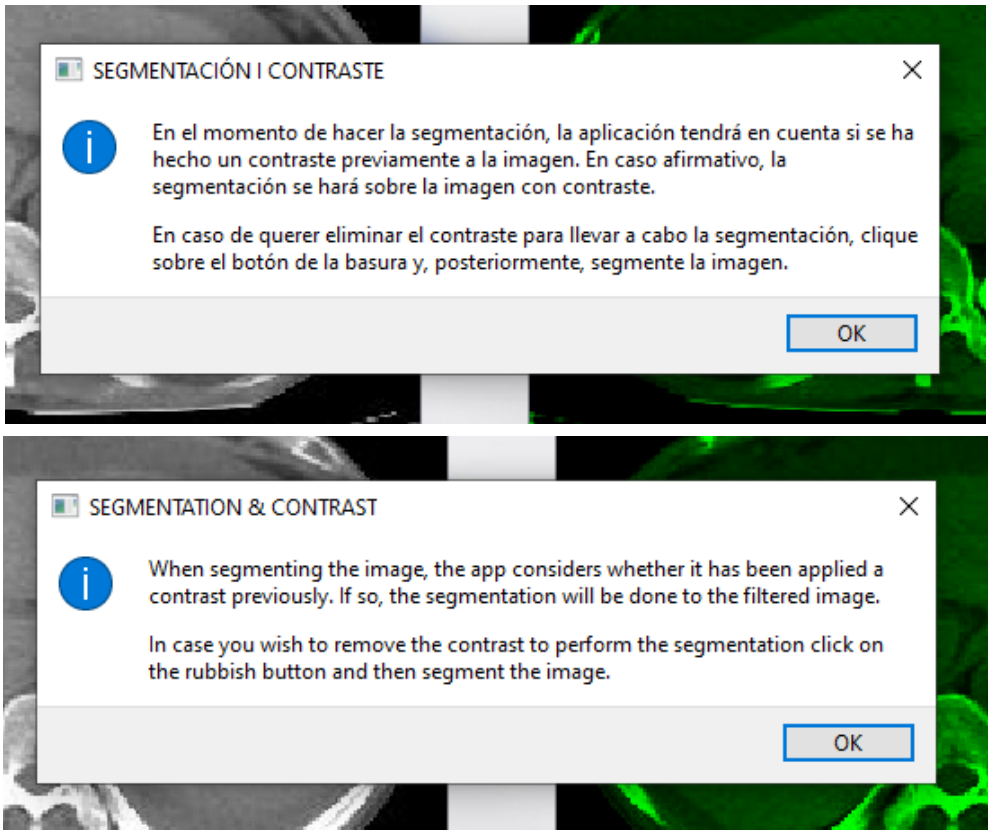

Figura 32: Exemple del canvi d'idioma tant al castellà com a l'anglès a les finestres emergents. En aquest cas és una finestra d'ajuda sobre el contrast.

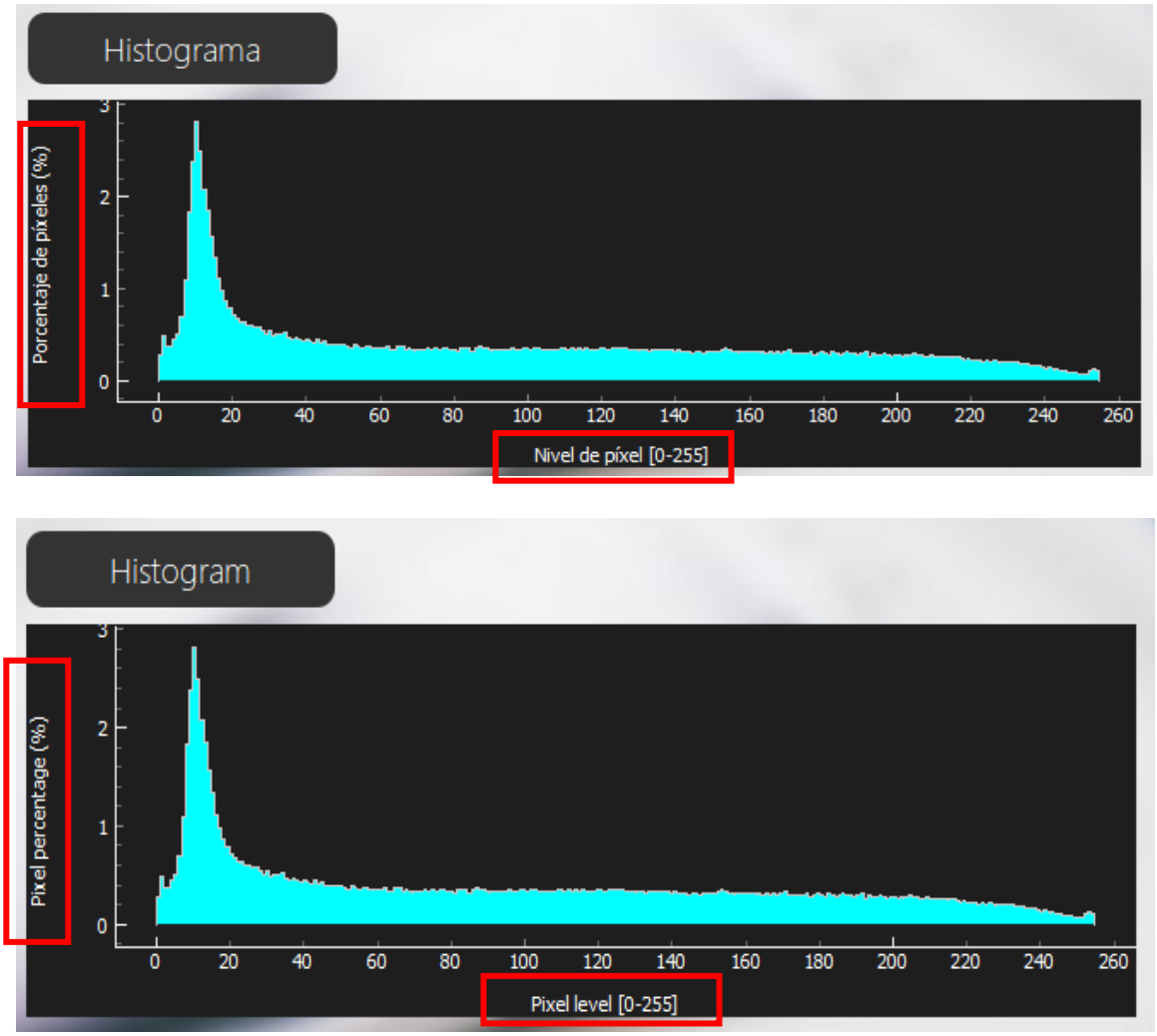

Figura 33: Exemple del canvi d'idioma tant al castellà com a l'anglès a widgets que mostren text (histograma i taula de propietats). En vermell s'assenyala el text del widget.

5.5. Llibreries utilitzades

5.5.1. Llibreries del software

Durant el projecte es van utilitzar una sèrie de llibreries de Python per a la manipulació de dades. Aquestes es van mantenir durant la segona part del projecte, i es van afegir algunes de noves per a la correcta complementació de les segmentacions. A continuació s'adjunta una llista de les llibreries necessàries, les funcions utilitzades i la utilitat de cadascuna per cada classe.

CLASSE	LLIBRERIA	FUNCIÓ	UTILITAT
SEGMENTADA	Scikit imatge	io	La funció io definida dins de skimage s'utilitza per a llegir les imatges, canviant el format jpg a format np.ndarray (una matriu de numpy)
		threshold_otsu	La funció threshold_otsu situada en el mòdul filters retorna el valor de llindar d'intensitat de la imatge basada en el mètode Otsu.
		sobel	La funció sobel situada en el mòdul filters detecta les vores dels objectes aplicar el filtre sobel.
		Clear_border	La funció clear_borders situada en el mòdul segmentation elimina artefactes lligats a les voreres dels objectes trobats en la imatge segmentada
		felzenszwalb	La funció felzenswalb situada en el mòdul segmentation calcula la manera més eficient de segmentar una imatge basada en els gràfics de Felzenszwalb.
		slic	La funció slic situada en el mòdul segmentation permet segmentar les imatges a partir de la classificació per k-means.

SEGMENTADA	Scikit image	closing	La funció closing situada en el mòdul morphology fa un tancament morfològic, una dilatació seguida d'una erosió, per tal d'esborrar el soroll anomenat pebre que hi pugui haver en la imatge.
		label	La funció label situada en el mòdul measures etiqueta les regions connectades entre si en una imatge en forma de matriu.
		watershed	La funció watershed situada en el mòdul segmentation segmenta la imatge amb la tècnica de transformació divisòria, en la qual s'interpreta la imatge en escala de grisos com una imatge topografica on el blanc es el punt més alt i el negre el més baix.
		label2rgb	La funció label2rgb en el mòdul color transforma una imatge d'etiquetes en una imatge a color per distingir els diferents elements.
		rgb2gray	La funció rgb2gray en el mòdul color transforma una imatge a color en una imatge a escala de grisos.
		square	La funció square situada en el mòdul morphology crea una matriu quadrada que és fa convolucionar en la funció closing
		resize	La funció resize, situada en el mòdul transform, redimensiona la imatge fent-la coincidir amb el valor que s'indica

SEGMENTADA	Scikit image	disk	La funció disk genera un element estructural pla amb forma de disc, de la mida desitjada.
		Median	La funció median genera un filtre mitjana multi-dimensional.
		gaussian	La funció gaussian genera un filtre gaussià multi-dimensional.
		regionprops	La funció regionprops mesura les propietats d'una imatge etiquetada per regions
		exposure	La funció exposure aplica una correcció gamma o logarítmica a la imatge d'entrada.
		KMeans	La funció KMeans aplica una segmentació K-Means on l'usuari introdueix el número de clústers.
		mixture.GaussianMixture	La funció mixture representa un model de distribució de probabilitats d'un model de barreja gaussiana. També permet estimar els paràmetres d'una distribució de barreja gaussiana. Es va utilitzar per calcular el GMM i el BIC per a cada número de clústers.
	NumPy		NumPy és el paquet fonamental per a la computació científica amb Python. Conté entre altres coses un potent objecte de matriu N-dimensional, funcions sofisticades (de difusió) i àlgebra lineal útil, transformació

SEGMENTADA			de Fourier i capacitats numèriques aleatòries.
	Matplotlib		Matplotlib és una biblioteca completa per crear visualitzacions estàtiques, animades i interactives a Python.
	Pandas		Pandas és una biblioteca de codi obert, amb llicència BSD, que proporciona estructures de dades de gran rendiment i fàcil d'utilitzar i eines d'anàlisi de dades per al llenguatge de programació Python.
	scipy	distance	La funció distance calcula distàncies paral·leles entre les observacions en l'espai n-dimensional. Es va utilitzar per calcular les distàncies entre els centroides de les gaussianes al GMM.
		stats	El mòdul stats conté un gran nombre de distribucions de probabilitats i una biblioteca creixent de funcions estadístiques. En concret es va utilitzar per a generar el plot de l'histograma amb gaussianes.
UI_TFG	datetime		El mòdul datetime proporciona classes per a la manipulació de dates i hores de maneres senzilles i complexes.

Taula 2: Llibreries i funcions de Python utilitzades al software de l'aplicació. [64]

5.5.2. Llibreries de la GUI

És important tenir present el canvi d'eines que s'han utilitzat a la interfície d'usuari. El més significatiu ha estat la substitució d'eines de Matplotlib per mostrar les imatges, el que ha comportat un canvi en la gran majoria de widgets restants.

Per a la programació de display, es va trobar molt més directe i eficient la utilització d'eines pròpies de Qt, que estan dirigides al ús en interfícies d'usuari. La utilització de Matplotlib implica la creació de varies classes per tal de representar el plot en la pantalla principal d'una GUI. En conseqüència, s'ha prescindit de totes aquelles classes que durant la primera etapa del programa eren necessàries (scene, plot_canvas, central widget, pandas_model i table widget), el que ha implicat una millora notable de temps d'execució i estalvi de codi.

Seguidament es mostra un llistat dels widgets que s'han utilitzat per a cada funció, en comparació amb els que s'utilitzaven en el primer període i els beneficis que han aportat aquests canvis:

	1a ETAPA	**2a ETAPA**	**BENEFICIS**
Disseny QMainWindow	Programant cada widget un a un	QtDesigner	• Permet compondre i personalitzar les finestres o els diàlegs de manera what-you-see-is-what-you-get (WYSIWYG) i provar-los utilitzant diferents estils i resolucions sense haver d'executar el programa cada cop. • Totes les propietats definides a Qt Designer es poden canviar dinàmicament dins del codi
Display imatge original i processada	Matplotlib	QgraphicsView	• Evitar la creació de classes complementàries. • Codi més senzill i directe • QgraphicsView permet fer zoom i implementar accions amb el ratolí, com veure les coordenades del píxel.
Histograma	Matplotlib	Pyqtgraph: WidgetPlot	• A diferència de Matplotlib, Pyqtgraph està pensat per a usos en aplicacions d'adquisició i anàlisi de dades • .Mentre que Matplotlib resulta més intuïtiu als programadors de Matlab, Pyqtgraph ho és pels programadors de Python o Qt • Execució molt més ràpida [65]
ToolBox	Matplotlib, automàtic	Manual, QLabel	• Al prescindir de Matplotlib, la Toolbox que automàticament apareix en la creació de qualsevol plot es va haver de crear funció a funció. Gràcies a les facilitats de QGraphisView es van poder implementar fàcilment.

			• Mitjançant un QLabel es mostren les coordenades de píxel.
Gràfic	Matplotlib	Matplotlib	• Al aparèixer en una finestra alternativa a la pantalla principal de l'aplicació no necessita de classes de suport. En aquest cas és més directe i senzill continuar amb Matplotlib.
Pop-up	No hi havia	QmessageBox	• Widget predeterminat per a aquesta acció. Permet el canvi de la icona i del text a mostrar.
Obrir imatge del PC	QfileDialog	QfileDialog	• Widget predeterminat per a aquesta acció. • Permet mostrar la imatge directament.
Taula de propietats de la imatge segmentada	Matplotlib	QtableWidget	• Evitar la creació de classes complementàries. • Codi més senzill i directe • Es pot canviar l'aparença fàcilment.

Taula 3: Widgets que s'han utilitzat per a cada funció de la interfície d'usuari en comparació amb el primer període juntament amb els beneficis que ha aportat els canvis.

6. Algorismes de segmentació

6.1. Comparació de tècniques de segmentació

A l'apartat ***¡Error! No se encuentra el origen de la referencia.*** del marc teòric de la memòria s'ha descrit i comparat les diferents tècniques de segmentació classificades en categories. Seguidament, es procedirà a la descripció de les segmentacions escollides per formar part de l'aplicació i la justificació de la elecció d'aquestes en concret.

A l'apartat ***¡Error! No se encuentra el origen de la referencia.*** s'ha considerat la implementació d'algunes de les tècniques no utilitzades.

CATEGORIA	SUB-CATEGORIA	SEGMENTACIÓ	JUSTIFICACIÓ
BASADES EN PÍXELS	Segons Llindar / Histograma	Otsu	• Es va voler incloure una tècnica de segmentació senzilla i tradicional per a imatges simples.
	Clústering	GMM	• Aquestes tècniques son no-supervisades i necessiten menys temps per generar els resultats [39], [40]. • Ofereixen una gran varietat en la metodologia per a diferents segmentacions, ja que trobem diversos subgrups; procediments basats en partició, en distribució i en densitat [41]. • Després de comprovar la eficiència de la segmentació K-Means durant el primer període del projecte, es va impulsar l'ampliació de l'espectre de tècniques de clustering, oferint segmentacions amb resultats eficients contrastats i heterogenis simultàniament.
		Quickshift	
		K - Means automàtic	
		K - Means manual	

	Xarxes neuronals	*No utilitzada*	• Precisen de molt temps per entrenar la xarxa [42]. • És una tècnica adequada per aplicar-la a tipus concrets d'imatges, però no quan les imatges són de naturalesa variada [42], com és el nostre cas.
BASADES EN VORES	Locals	*No utilitzada*	• Aquestes tècniques només són eficients per imatges simples i sense soroll [66]. L'aplicació ja disposa d'un tipus de segmentació senzill (Otsu). • No s'adapten automàticament a les variacions de cada regió, sinó que necessiten ser ajustades inclús per cada tipus d'imatge [44]. • No poden segmentar la imatge per sí mateixes, tot i que proporcionen informació sobre els contorns que pot ser útil i utilitzada en combinació amb altres [45]. • Moltes de les fronteres localitzades pateixen de discontinuïtats o sobre-deteccions, el que fa que els resultats només siguin *candidats* a vores reals. • Aquest tipus de tècniques són les menys recomanables.
	Globals		
	Models deformables		
	Fusió i / o divisió	*No utilitzada*	• Tendeixen a resultar en contorns poc realistes i abruptes [29].

BASADES EN REGIONS	Morfològiques	Watershed	• És la tècnica basada en regions més representativa [49], [50]. • Té l'avantatge de proporcionar vores contínues i tancades. • Tendeix a veure's menys afectada pel soroll de la imatge
	Creixement de regions	*No utilitzada*	• Mitjançant aquesta tècnica, cada llavor dona lloc a una regió, és a dir, la imatge es divideix en tantes regions com llavors es col·loquen al principi [46], [48], [52]. Això crea una forta dependència en els coneixements sobre segmentació per part de l'usuari, que es poden garantir.

Taula 4: Taula comparativa dels tipus de segmentacions existents amb la justificació d'haver-les o no escollit per a formar part de l'aplicació.

S'ha de tenir en compte que a causa de la naturalesa de l'aplicació s'han considerat únicament les tècniques de classificació no supervisades, ja que són les que encaixen amb el perfil d'usuari. Com a conseqüència, s'han descartat mètodes de segmentació basats en Anàlisi discriminant linear (LDA), Xarxes neuronals convolucionals (CNNs) i anàlisi de components principals (PCA) amb màquines de suport vectorial (Eigenfaces).

La implementació d'aquestes tècniques va implicar diferents graus de dificultat. Mentre que la segmentació Quickshift disposava d'una funció pròpia de Python que calcula la segmentació automàticament, les segmentacions de K-Means i Gaussian Mixture van haver de ser elaborades manualment. La primera no va suposar una gran diferència respecte a l'automàtica, contràriament, pel Gaussian Mixture Model es va haver de refinar el mètode automàtic propi de *Scikit Learn*, ja que el resultat en primera instància no era òptim. És per aquest motiu que es fa especial menció a aquesta tècnica a continuació.

6.2. Algorisme K-Means

L'algorisme K-Means opera sobre un set de punts de dades tot assumint que el nombre de grups o clústers (K) presents a la imatge és conegut. En el cas de K-Means automàtic aquest nombre s'escull aleatòriament, mentre que en la funció implementada K-Means manual és l'usuari qui la introdueix. Per a segmentacions d'imatges el set de dades es correspon amb els valors de píxel. En un inici l'algoritme escull K punts aleatoris a la imatge i els assigna com a centres o centroides dels clústers. Aleshores, es forma la primera distribució de clústers associant cada píxel al clúster de centroide més proper. Seguidament, aquests dos passos es segueixen fins a la convergència:

1- Es tornen a computar els centres de cada clúster mitjançant la mitjana de tots els valors de píxel que formen el clúster.
2- Cada punt de dades es reassigna al clúster amb centre més proper. La distribució de clústers queda formada altre cop.

La convergència s'assumeix quan no calen més reassignacions, és a dir, quan la distribució de clústers no canvia respecte la anterior iteració. La convergència està garantida en un número finit de iteracions, no obstant tendeix a un òptim local, depenent molt del número de clústers K que es decideixen en un inici [67].

6.3. Algorisme Gaussian Mixture Model (GMM)

Es va dedicar una gran part del temps d'elaboració del projecte al desenvolupament d'un tipus de segmentació més sofisticada pels casos més exigents, mitjançant l'agrupament amb Gaussian Mixture Model.

Seguidament s'explica amb detall la elaboració tant de la funció de segmentació per barreja de gaussianes per imatges en escala de grisos com la mateixa per imatges RGB.

6.3.1. Funció GaussianMixture de Scikit-Learn

La funció *GaussianMixture* de Scikit-Learn és la que s'ha utilitzat al clústering de la segmentació per GMM. Abans d'explicar el procediment que s'ha seguit per refinar aquesta funció i adaptar-la al tipus d'imatges que es processen a l'aplicació, cal ser conscient de les principals característiques, habilitats i limitacions de la mateixa.

6.3.1.1. Metodologia de l'algorisme

De la mateixa manera que amb el cas de K-Means, l'algorisme per mescla de gaussianes també assumeix un número de clústers conegut (K), la diferència és que en aquest cas es precisa d'un mètode per determinar-lo. Es va escollir el criteri d'informació Bayesià per a aquesta finalitat, ja que Scikit Learn proporciona una funció dins de *GaussianMixture* per calcular-lo.

La distribució de valors de píxel doncs es modelitza com a una suma ponderada de gaussianes amb pesos Π_k i cadascuna amb una mitjana μ_k i una desviació típica σ_k de la següent manera:

$$P(x) = \sum_{k=1}^{k} \Pi_k N(\mu_k, \sigma_k)$$

Equació 1: forma de les distribucions gaussianes que formen el model de barreja de gaussianes [60].

Els pesos han de tenir valors positius i la seva suma ha de donar 1. La funció *GaussianMixture* determina els paràmetres de cada component del model (mitjana, covariància i pes) mitjançant l'algorisme de maximització d'expectatives [57]. Finalment, cada distribució Gaussiana equival a un clúster, i per a cada píxel es determina la probabilitat de pertànyer a cadascun dels clústers.

6.3.1.2. Funcions fit i predict

La funció *GaussianMixture.fit* proporciona un mètode per aprendre un model de barreja gaussiana a partir de dades d'entrenament. Després, mitjançant la funció *GaussianMixture.predict* es testeja la imatge assignant a cada mostra la distribució gaussiana a la que probablement pertany [57]. En el nostre cas las dades d'entrenament i de test provenen de la mateixa imatge (la imatge seleccionada per l'usuari) ja que al tractar amb tot tipus d'imatges diferents no s'ha optat per entrenar al mètode amb una base de dades d'imatges d'un sol tipus. Però, aquesta característica segueix representant un fet important ja que donat el cas que, tal com ha succeït durant el desenvolupament del projecte, es trobin dades a la imatge que distorsionin els resultats i no aportin informació (es veurà a continuació que els píxels extrems de la imatge en són un exemple) es pot entrenar el model amb una versió de la imatge obviant aquestes dades irrellevants i després testejant-lo amb la imatge original.

6.3.1.3. Dades d'entrada i de sortida

Un fet que també gaudirà d'importància als següents capítols de la memòria és que la funció *GaussianMixture* té la capacitat d'acceptar com a paràmetres d'entrada, a part del número de clústers de la imatge, el pes, la mitjana i la covariància que ha de tenir cada distribució gaussiana del model [57]. Això implica que, mitjançant la funció, un cop extrets els resultats de les distribucions que teòricament formen el model més exacte amb els seus paràmetres, aquestes es poden modificar i

filtrar per després donar com a dades d'entrada les que resultin convenients. És a dir, donat el cas que la funció retorni un número de clústers teòricament idoni *X* però mitjançant altres procediments es confirmi que una de les distribucions és prescindible, es podrà donar com a dades d'entrada no només el número de clústers (*X – 1)*, sinó també les mitjanes de les distribucions que s'han confirmat que sí són rellevants, amb la finalitat que la funció *GaussianMixture* recalculi únicament paràmetres restants com poden ser el pes de cada distribució al model.

6.3.2. Algorisme per imatges en escala de grisos

En primer instància es va tractar d'utilitzar la funció directament sobre una imatge senzilla, és a dir, imatge l'histograma de la qual tingués formes de gaussiana ben definides. El resultat però, era ambigu. Mentre que la imatge resultant era acceptable donat la variabilitat de tonalitats de cada zona, el número de gaussianes que es va trobar va ser significativament superior al que s'hauria esperat, ja que l'histograma mostrava una clara tendència a 4 gaussianes (3 taques de color més el color blanc del fons), i el programa en va retornar 8.

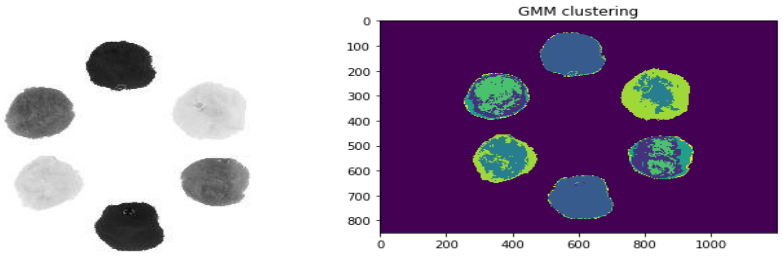

Figura 34: Primera imatge de prova en grayscale i resultat de la segmentació, respectivament.

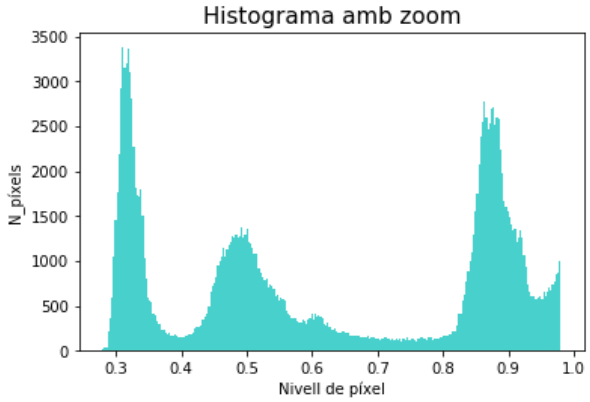

Figura 35: Histograma de la imatge d'entrada entre els valors [0.2-0.9] per apreciar la tendència amb forma gaussiana.

L'excés de precisió en reconeixement d'objectes és relatiu. Si estem analitzant un teixit teòricament uniforme en busca d'un tumor, es necessita que el programa trobi diferències amb la major exactitud possible. Però si en canvi s'intenta separar cèl·lules de diferents mides o formes, l'objectiu serà

classificar cèl·lules senceres com a un únic objecte. És va comprovar doncs si en el cas d'una imatge mèdica el nivell de detall amb el que trobava gaussianes era excessiu.

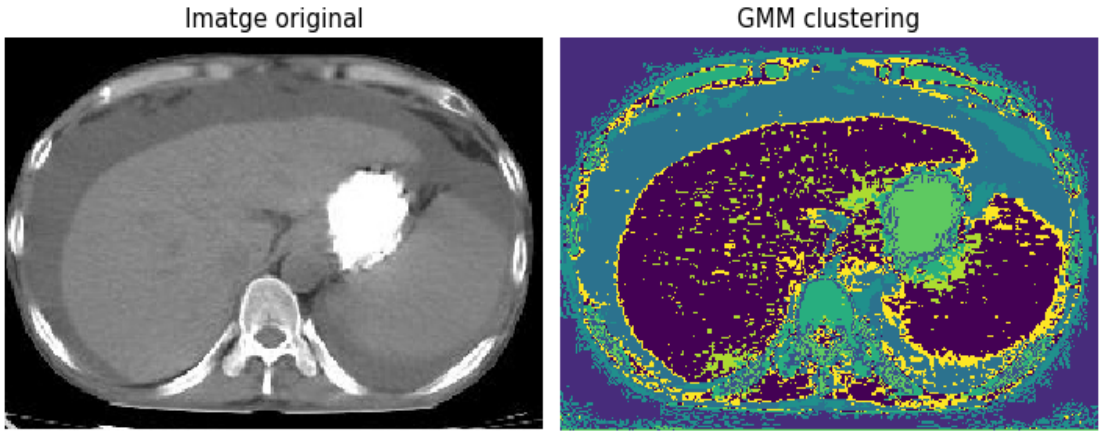

Figura 36: Exemple de segmentació d'una imatge mèdica amb la funció **GaussianMixture** *de* Scikit Learn. *A l'esquerra la imatge original i a la dreta la segmentada.*

Es va constatar una tendència al overfitting. Per buscar una forma d'emmotllar la funció de Pyhton a les nostres necessitats es va tenir en compte la informació que aquesta ofereix. Al executar la funció, no només retorna el número de gaussianes que defineixen la distribució, sinó que també indica la mitjana de cada gaussiana, les seves covariàncies (i, per defecte, les seves sigmes) i els pesos de cada gaussiana (probabilitat).

És per això que es van contemplar 4 mètodes diferents per solucionar-ho; aplicar un filtre passa baix a la imatge per suavitzar els canvis de tonalitat bruscs, estudiar la variació del vector BIC, eliminar els píxels extrems i filtrar el número de gaussianes segons la seva probabilitat.

6.3.2.1. Filtre passa baix

La idea és filtrar la imatge original amb un filtre passa baix abans de transformar-la en vector. El filtre passa baix farà que els píxels aïllats que tenen una tonalitat anormal en la seva zona es difuminin adoptant el nivell dels píxels del voltant. D'aquesta manera s'evita que es considerin com a rellevants. Existeixen principalment dos tipus de filtre passa baix; filtre gaussià i filtre mitjana.

- **Filtre gaussià**

  El paràmetre que defineix la intensitat del filtre és el valor de sigma. Així, quant més alt és aquest valor més borrosa serà la imatge resultant. A priori es va observar l'efecte d'un filtre gaussià amb valors de sigma 1, 5 i 10.

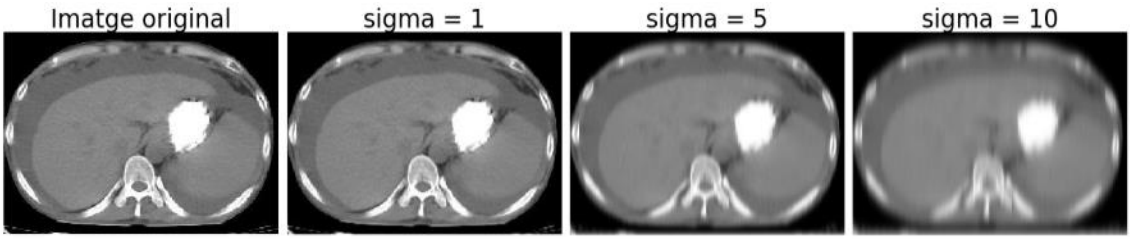

Figura 37: Imatge original i amb filtre gaussià aplicat amb valors de sigma d' 1, 5 i 10 respectivament.

Donat el fet que el resultat desitjat implica la conservació dels contorns i la eliminació del soroll de *salt and pepper,* es va concloure que una sigma de valor 10 distorsionava exageradament la imatge, i es va optar per descartar-la. Es van obtenir els següents resultats de la segmentació de les altres dues imatges filtrades:

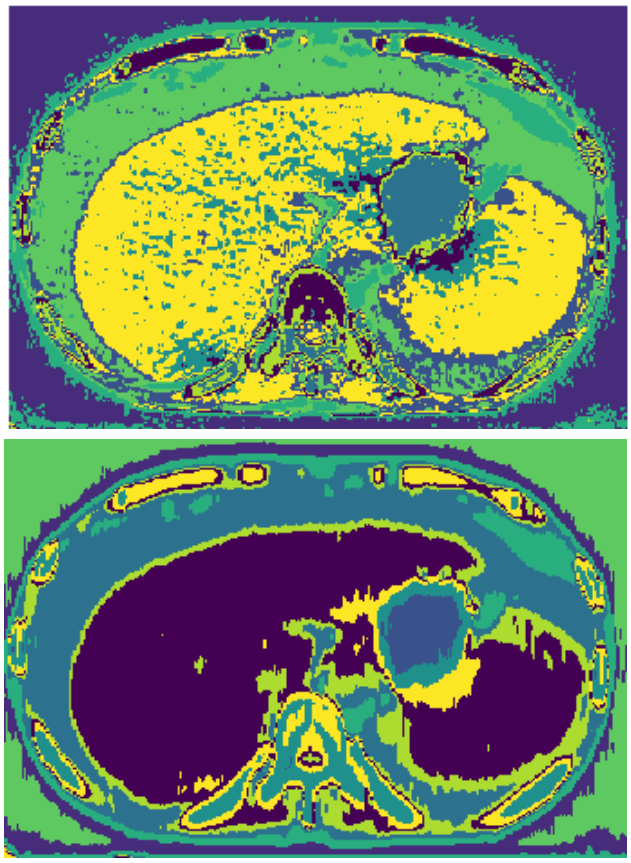

Figura 38: Comparació d' imatge segmentada amb filtre gaussià per sigma amb valor 1 i 5, respectivament.

- **Filtre mitjana**

En aquest cas el factor decisiu és la mida del disk. Tornem a provar amb valors de 2, 5 i 10.

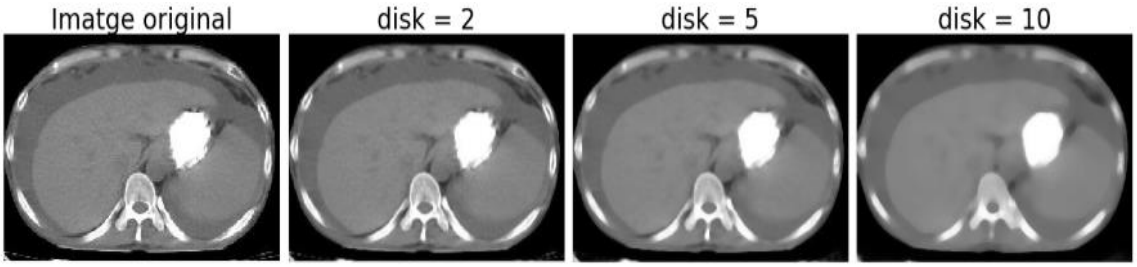

Figura 39: Imatge original i amb filtre mitjana aplicat amb disks de mida 2, 5 i 10 respectivament.

Altre cop el disk més petit no suavitza suficient la imatge i el més gran provoca una pèrdua clara d'informació, és per això que es va segmentar la imatge amb filtre mitjana de disk 5. El resultat d'aquesta es mostra a la Figura 40.

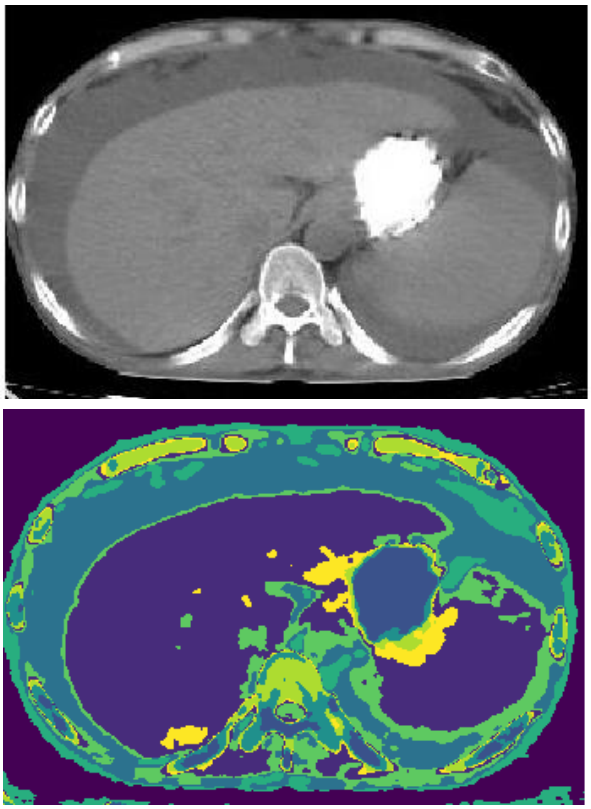

Figura 40: Comparació entre la imatge original i la segmentada amb la funció **GaussianMixture** *de* Scikit Learn *després d'haver aplicat un filtre mitjana amb disk de mida 5.*

- **Comparació d'ambdós filtres**

Després de fer les proves pertinents amb els dos tipus de filtre es va observar una millora al resultat, però sense l'assoliment total de la segmentació òptima. Un filtre suau no evitava l'overfitting, i filtres més severs difuminaven els límits de cada zona distintiva a la imatge fent que la segmentació no correspongués amb la realitat.

Es va concloure que el millor era aplicar un filtre mitjana i afegir altres variacions.

6.3.2.2. Eliminació dels píxels extrems

Un fet comú a la majoria de les imatges era que al histograma hi havia una gran quantitat de píxels de valor 0 i 255, normalment corresponent al fons. Al ser pics molt alts en comparació amb la resta de nivells, es va pensar que podien estar distorsionant la interpretació del programa per trobar les gaussianes. És per això que es va provar d'executar el programa amb un vector editat, on s'havien eliminat tots els píxels de nivells 0 i 255 i propers a aquests.

Òbviament aquest nou vector sense valors extrems va ser només utilitzat durant l'entrenament del programa (*train),* mentre que, un cop el programa tenia internalitzat quina forma i mida haurien de tenir les gaussianes, es va utilitzar el vector original per fer la segmentació real (*fit).*

El resultat va ser el següent:

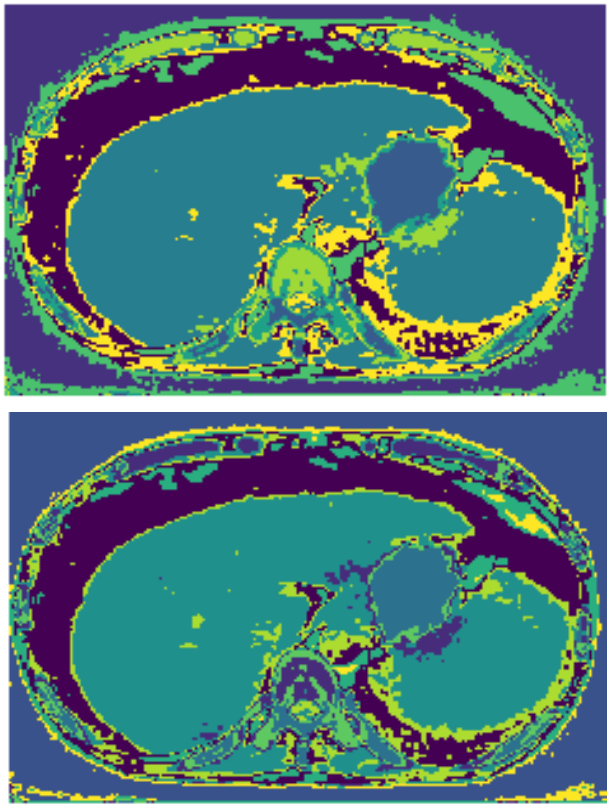

Figura 41: Comparació del resultat de la segmentació entre l'aplicació únicament de la funció **GaussianMixture** *de* Scikit Learn *i el mateix procediment havent eliminat els píxels de nivells extrems primer, respectivament.*

Es va trobar que hi havia una millora en la rellevància dels detalls trobats pel programa, tot disminuint el soroll.

6.3.2.3. Gradient del vector BIC

El vector BIC presenta sempre la mateixa forma: un decreixement molt ràpid al principi seguit d'una monotonia (Figura 42). Tot i que la funció BIC té en compte que l'augment del número de clústers està restringit a una millora en la qualitat de la segmentació, aquest encara tendia a trobar un valor òptim massa elevat. Es va provar de calcular el vector BIC fins a 25 gaussianes, tot trobant l'òptim a 22 clusters (Figura 43) . Es va pensar doncs que una possible solució era filtrar aquest vector, condicionant l'augment de gaussianes amb que el valor del BIC disminuís com a mínim en un x% respecte l'anterior. És a dir, afegir gaussianes només si aporten una millora notable a la segmentació, per estalviar-nos les que no aporten informació útil.

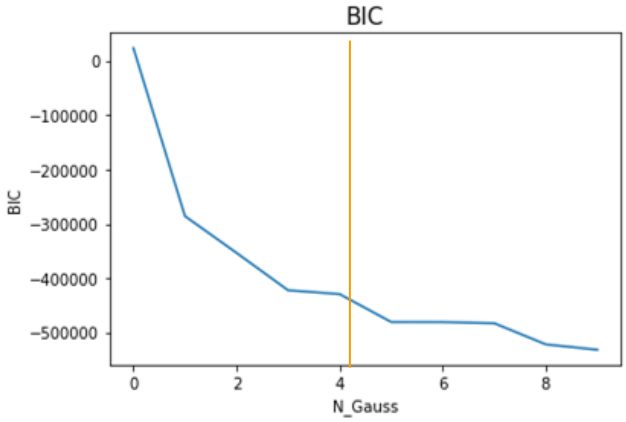

Figura 42: Representació gràfica del vector BIC calculat a través d'una segmentació per Barreja de Gaussianes mostrant una clara tendència al decreixement en augmentar el número de gaussianes.

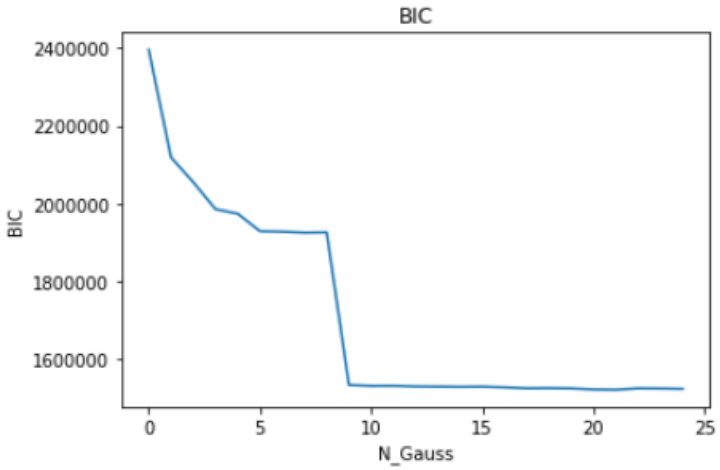

Figura 43: Representació gràfica del vector BIC calculat a través d'una segmentació per Barreja de Gaussianes per un total de 25 gaussianes.

Un altre problema va ser el de decidir el valor del percentatge de millora a considerar acceptable (tolerància), tot sent un paràmetre variable i que òbviament condicionava l'èxit del programa. Després de diferents proves, es va establir un valor estàndard del 10%.

Al provar-ho amb diferents imatges es va trobar que el valor de tolerància havia de ser modificat a mà contínuament, el que implicava que el mètode patia d'una alta variabilitat. Donat el fet que la segmentació havia de formar part d'una GUI on els usuaris carien de coneixements de les funcions internes del programa, es va optar per seguir investigant alternatives.

6.3.2.4. Discriminació de gaussianes segons proximitat

- **Base teòrica del procediment**

Aquest mètode es basa en el fet que a la funció mixture model amb la que estem treballant accepta com a paràmetre d'entrada les mitjanes de les gaussianes que ha de trobar. Per establir una metodologia concreta es va tenir en compte la següent informació:

a) Per establir els pesos de cada gaussiana, la funció té en compte tant el número de píxels que formen aquell nivell de píxel com els que formen els nivells de píxel del voltant.

   Veiem el següent exemple:

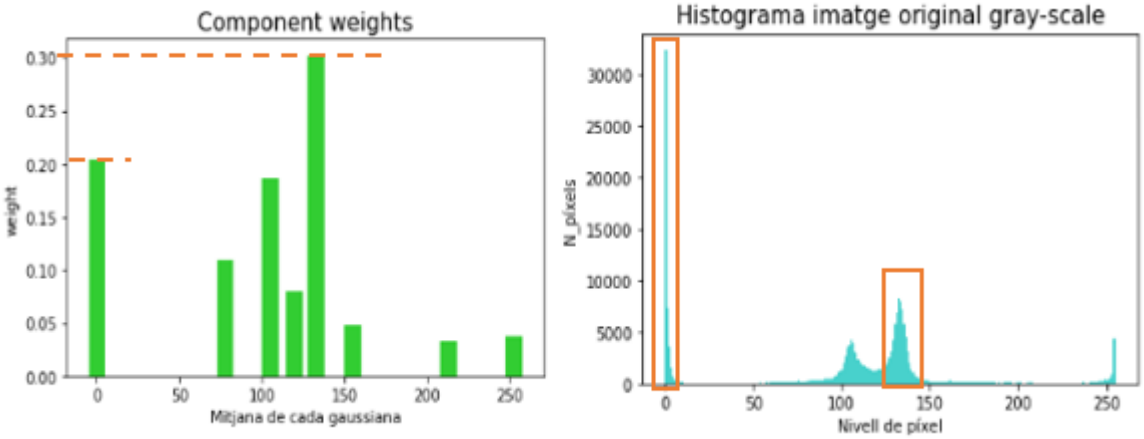

Figura 44: Plot dels pesos que tenen cada gaussiana col·locats a la seva mitjana corresponent a l'esquerra, i histograma de la imatge a la dreta.

   Veiem l'histograma de la imatge en contraposició amb els pesos que se li ha assignat a cada gaussiana (els pesos estan col·locats a l'eix x a la posició de la mitjana de la seva gaussiana corresponent). Mentre que el número de píxels de nivell 0 és molt superior als del voltant del nivell 140, el pes atorgat a la gaussiana d'aquest segon és superior. Això es deu a que la probabilitat de que un píxel formi part d'un determinat clúster és proporcional tant al número de píxels d'aquell nivell com dels del seu voltant.

b) La discriminació de gaussianes no es pot fer en funció de la probabilitat de cadascuna, ja que gaussianes amb poca probabilitat (com la dels píxels de nivell 255, blanc) aporten una informació indispensable.

c) Ens basem doncs en el fet que cap histograma té forma de gaussiana perfecte. Al haver-hi irregularitats que formen pics a l'histograma, el programa interpreta que hi ha varies gaussianes on només n'hi ha una (Figura 45 i Figura 46). Així doncs, l'indicador de que una gaussiana és prescindible serà la proximitat de la seva mitjana amb les gaussianes del voltant.

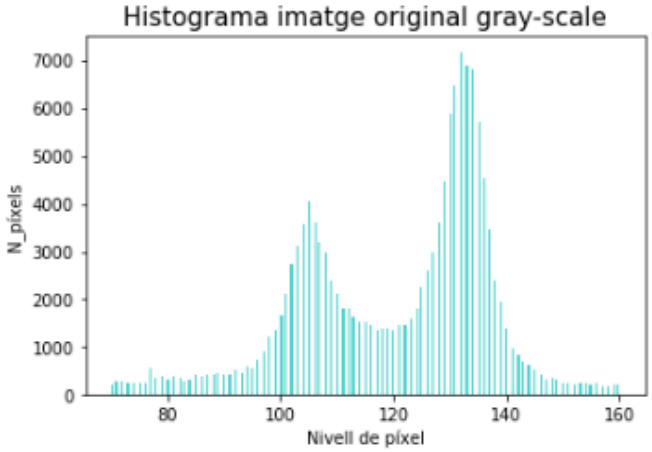

Figura 45: Histograma de la imatge al rang [70,160] on s'aprecien les imperfeccions en comparació amb una forma gaussiana.

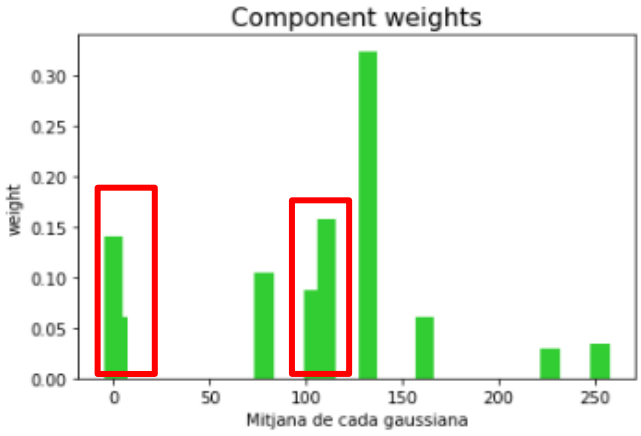

Figura 46: Exemple de mitjanes de gaussianes trobades per la funció **GaussianMixture** *de* Scikit Learn *que estan massa a prop entre sí per a tenir rellevància.*

- **Metodologia del procediment**
  1. Aplicació del filtre passa baix suau per reduir el soroll de sal i pebre a la imatge.
  2. Execució repetida de la funció Gaussian Mixture Model, augmentant a cada volta el número de components, en el rang [1,10] i càlcul del valor BIC.

3. Obtenció del valor mínim de BIC.
4. *Fit* del Gaussian Mixture Model amb número de gaussianes òptim i obtenció de les seves variables.
5. Creació d'una matriu on la primera columna representa les mitjanes de cada gaussiana ordenades de menor a major, i la segona correspon als pesos de cadascuna, de manera:

$$\begin{pmatrix} \mu_0 & \omega_0 \\ \mu_1 & \omega_1 \\ \dots & \dots \\ \mu_n & \omega_n \end{pmatrix}$$

Figura 47: Forma de la matriu, on μ representa les mitjanes i ω els pesos.

6. Eliminació de les gaussianes que estan massa properes (sempre eliminant la que té una probabilitat menor).
7. Execució de la funció gaussian mixture model introduint els valors trobats de número de components i mitjanes. El programa recalcula els pesos de cada gaussiana.
8. *Fit* amb el GMM establert.

El resultat del programa ara és el següent:

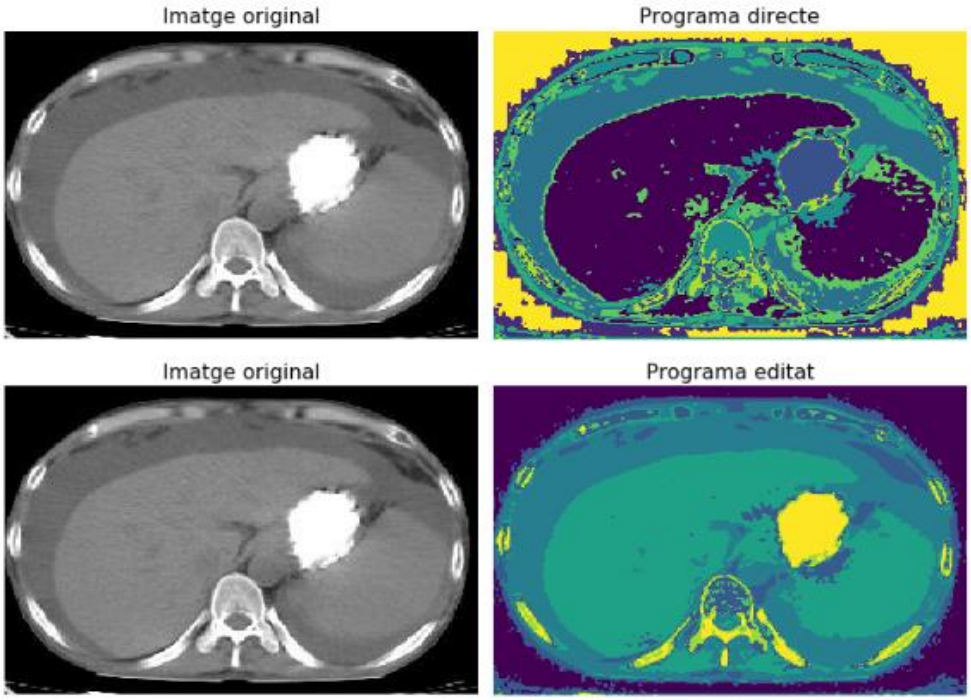

Figura 48: Comparació d'una segmentació aplicant únicament un filtre passa baix a la imatge abans de segmentar-la amb la funció **GaussianMixture** *de* Scikit Learn (imatge a dalt a la dreta) *i mitjançant el mètode de discriminació de gaussianes un cop segmentada la imatge amb la mateixa funció (imatge a baix a la dreta). Les dues imatges situades a l'esquerra corresponen a la imatge original.*

Finalment es va aconseguir el resultat desitjat. Per comprovar que el codi era immune a variacions es van testejar diferents imatges:

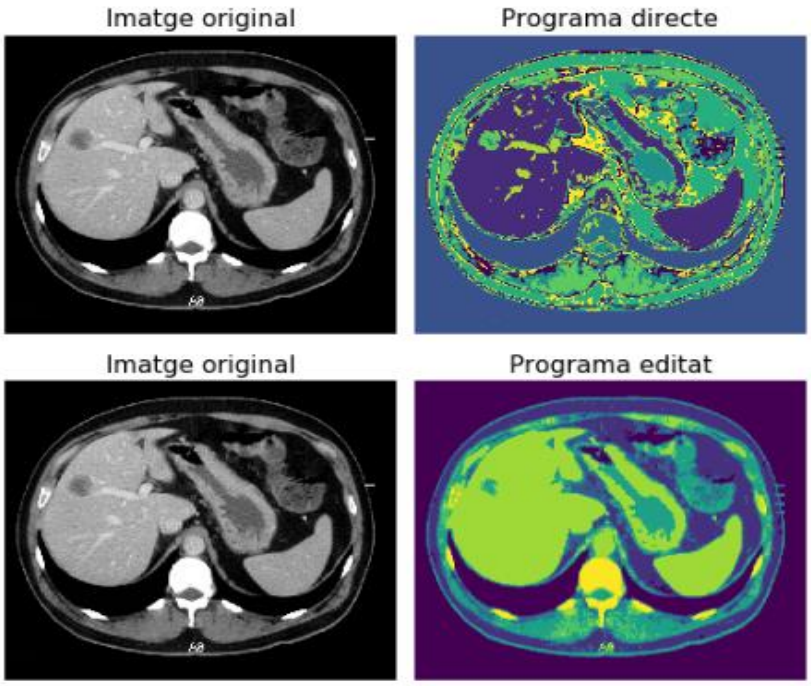

Figura 49: Comparació d'una segmentació aplicant únicament un filtre passa baix a la imatge abans de segmentar-la amb la funció **GaussianMixture** *de* Scikit Learn (imatge a dalt a la dreta) *i mitjançant el mètode de discriminació de gaussianes un cop segmentada la imatge amb la mateixa funció (imatge a baix a la dreta). Les dues imatges situades a l'esquerra corresponen a la imatge original.*

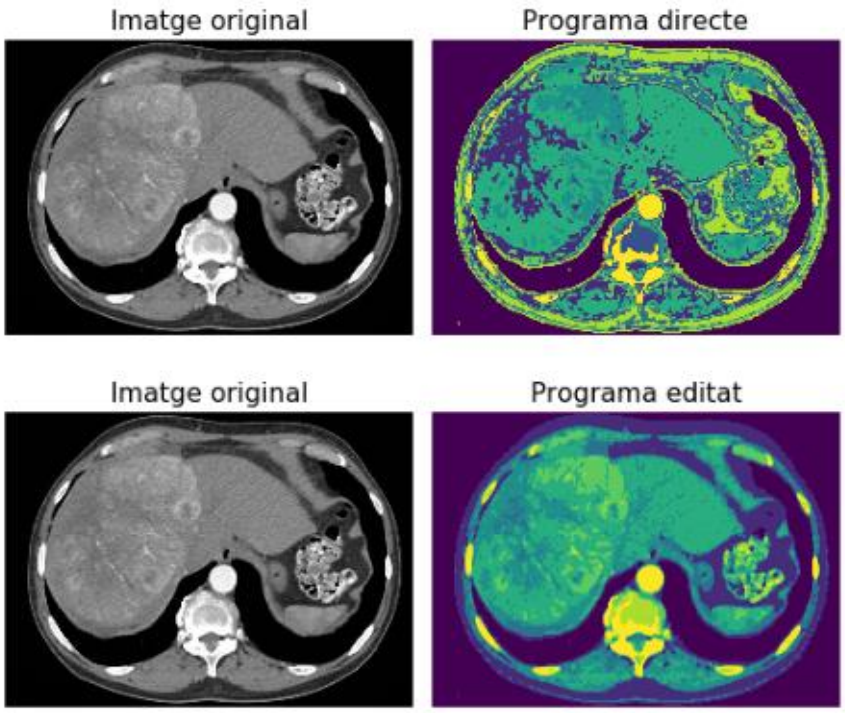

Figura 50: Comparació d'una segmentació aplicant únicament un filtre passa baix a la imatge abans de segmentar-la amb la funció **GaussianMixture** *de* Scikit Learn (imatge a dalt a la dreta) *i mitjançant el mètode de discriminació de gaussianes un cop segmentada la imatge amb la mateixa funció (imatge a baix a la dreta). Les dues imatges situades a l'esquerra corresponen a la imatge original.*

6.3.3. Algorismes per imatges RGB

Es va diferenciar el processament depenent de si la imatge original era en color o en escala de grisos, ja que el fet que les imatges RGB continguessin 3 canals implicava tenir en compte un paràmetre de la funció MixtureModel que fins ara no era rellevant: el tipus de matriu de covariància.

Per poder establir una comparació objectiva entre la funció de *Scikit Learn* i la funció editada es va escollir una imatge i es van parametritzar les zones que s'haurien de diferenciar en una segmentació òptima. La imatge de test doncs va ser la de la Figura 51. El resultat òptim es va establir com la diferenciació entre els objectes vermells (1), liles (2) , negres (3) i els contorns blancs (4). També havia de ser capaç d'identificar-los a la zona més fosca (5).

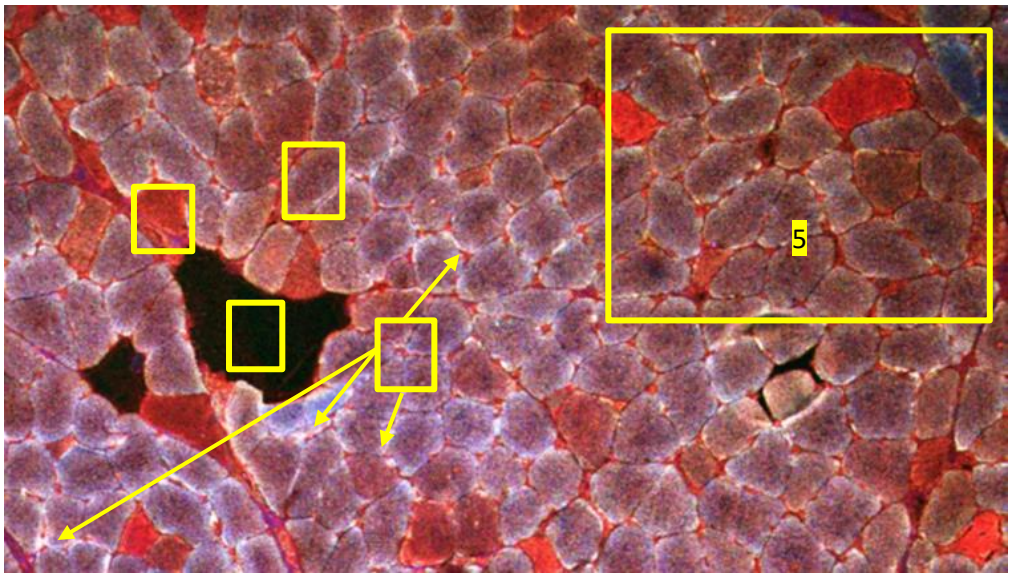

Figura 51: Representació de les regions que haurien de resultar clarament diferenciades després de la segmentació de la imatge per a validar-la..

6.3.3.1. Funció Mixture Model

Es va seguir el mateix procediment que amb les imatges en escala de grisos, tot executant en primer lloc el programa sense modificacions. Es va calcular el BIC no només per diferent número de clústers sinó també per cada tipus de matriu de covariància (spherical, tied, diagonal, full). Es va obtenir un resultat com el de la figura:

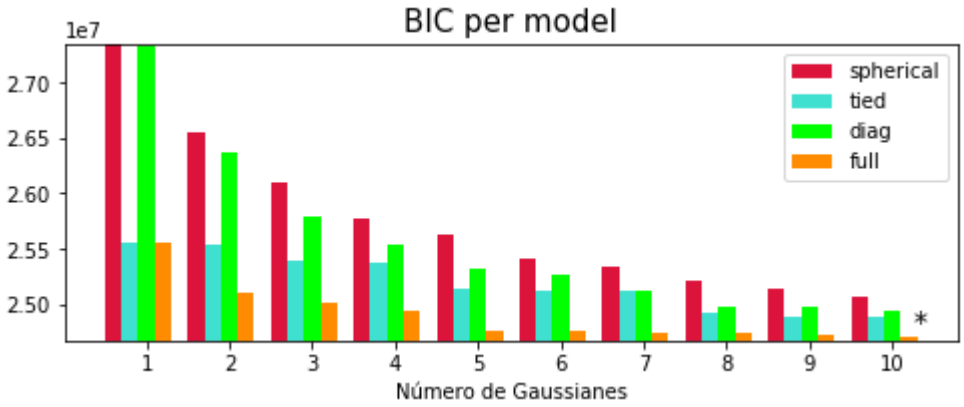

Figura 52: Representació gràfica del vector BIC resultat de la funció Gaussian Mixture Model per imatges en RGB.

Com era d'esperar, el vector BIC presentava un comportament similar al de la imatge grayscale, disminuint proporcionalment amb l'augment del número de gaussianes. De la mateixa manera es va observar que la millora conseqüent d'afegir gaussianes a partir de cinquena era massa lleugera.

Així doncs, la imatge segmentada va ésser:

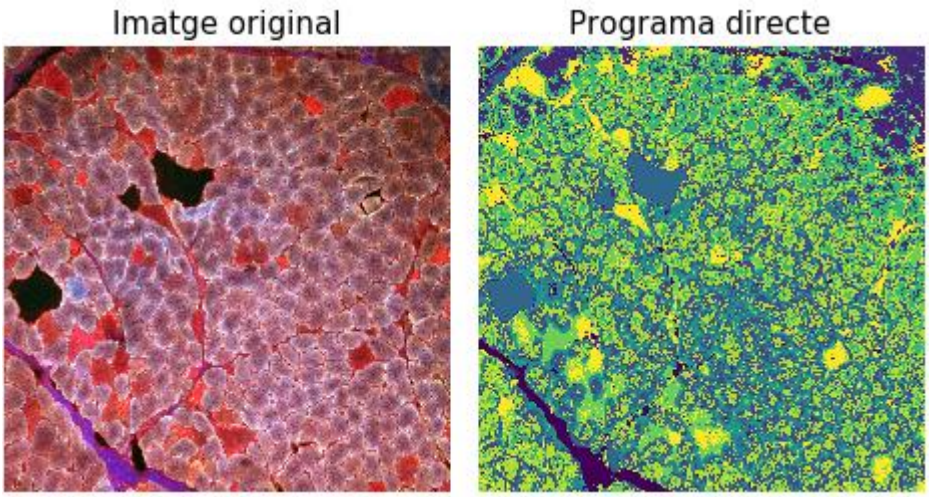

Figura 53: Comparació entre la imatge original i segmentada mitjançant la funció **GaussianMixture** *de* Scikit Learn.

Es va comprovar, tal com era d'esperar, que el programa tendia a un clar overfitting limitant l'apreciació òptima de cada objecte. Per aquest motiu es va procedir a l'adaptació de la funció per a les nostres necessitats, tot editant el programa mitjançant el procediment que havíem trobat més eficaç per imatges en escala de grisos.

6.3.3.2. Discriminació de gaussianes segons la proximitat

Al intentar obtenir els mateixos paràmetres que havien fet possible la discriminació de gaussianes es va trobar que el vector amb les mitjanes de cada gaussiana retornava tres coordenades enlloc d'una,

corresponents al centroide de la gaussiana en 3D, expressat com el nivell de píxel de cada canal (vermell, verd i blau, Figura 54).

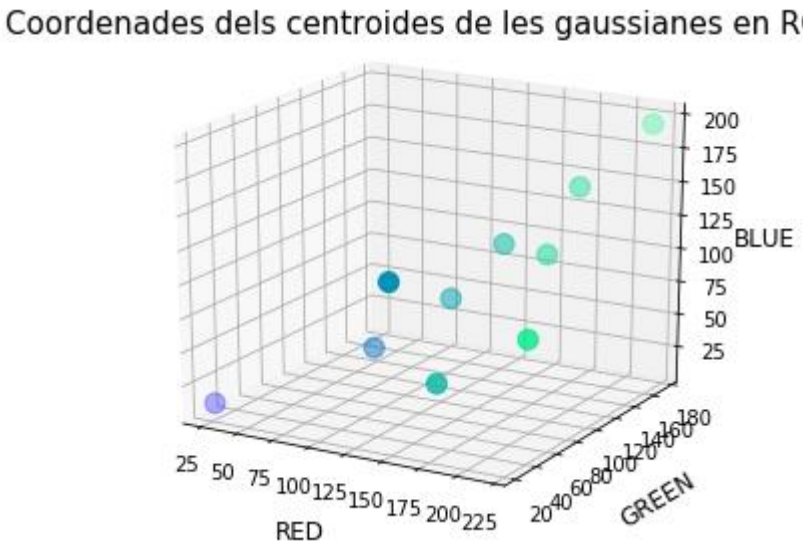

Figura 54: Representació de cada un dels centroides de les gaussianes trobades la funció MixtureModel.

$$Gauss_n = [Coord_R \;\; Coord_G \;\; Coord_B]$$

Equació 2: Forma del centroide per cada una de les n gaussianes

La implicació més important de la tridimensionalitat del paràmetre coordenades és el mètode pel qual es va calcular la proximitat entre gaussianes, enlloc de calcular la distància rectilínia es va fer mitjançant la distància euclidiana, de forma:

$$d(P,Q) = \sqrt{(p1-q1)^2 + (p2-q2)^2 + (p3-q3)^2} =$$

$$\sqrt{(R1-R2)^2 + (G1-G2)^2 + (B1-B2)^2}$$

Equació 3: Exemple d'equació utilitzada pel càlcul de la distància euclidiana entre les mitjanes de dues gaussianes. R *és la coordenada del centroide de la gaussiana en el canal red,* G *al canal green i* B *al canal blue, els subíndex* 1 *i* 2 *correspondrien a les gaussianes 1 i 2.*

Per poder fer el càlcul entre totes les gaussianes es va utilitzar la funció de Python *scipy.spatial.distance.cdist*.

Es va fer la discriminació de la mateixa manera que per imatges en escala de grisos. Finalment, reproduint un altre cop el que havíem fet per les imatges grayscale, es van introduir els nous valors de mitjanes al GMM per a que es calculessin els nous pesos de cada gaussiana.

El resultat obtingut va ser el següent:

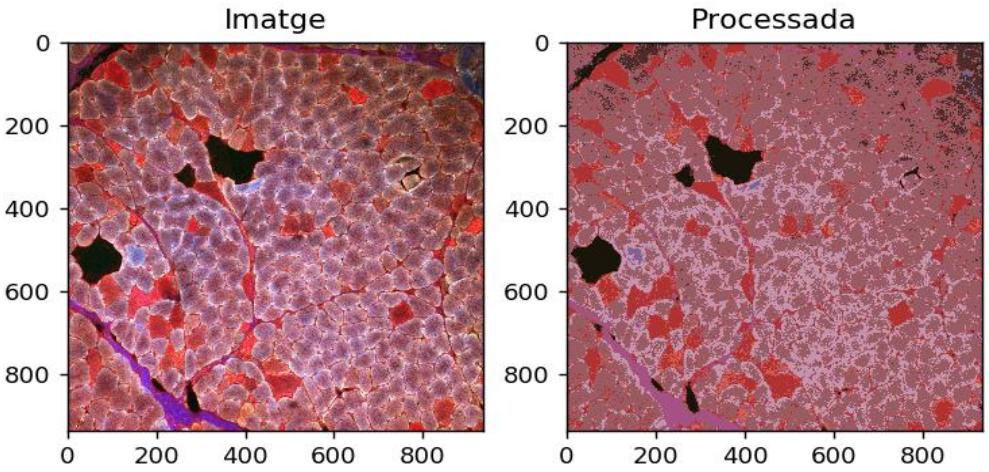

Figura 55: Exemple 1 del resultat de la segmentació mitjançant el mètode de discriminació de gaussianes. A l'esquerra la imatge original i a la dreta la processada.

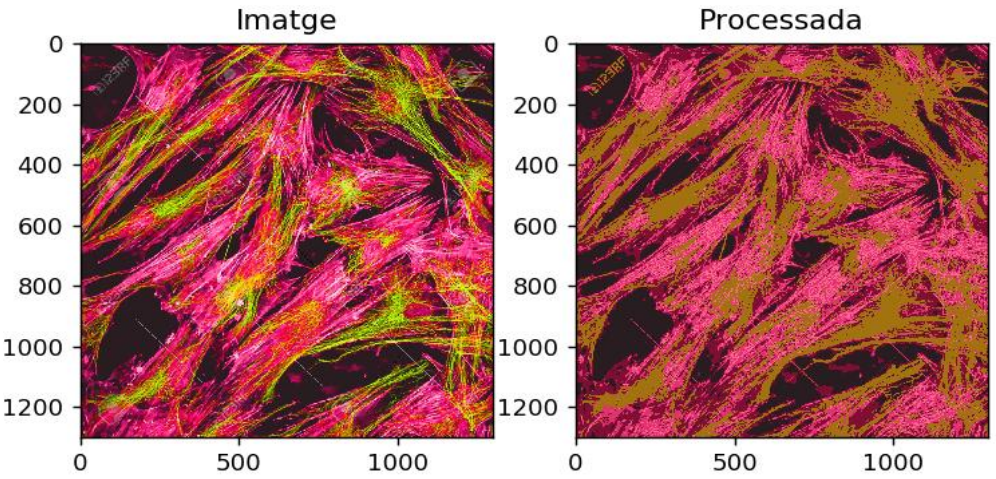

Figura 56: Exemple 2 del resultat de la segmentació mitjançant el mètode de discriminació de gaussianes. A l'esquerra la imatge original i a la dreta la processada.

6.3.4. Histograma amb les gaussianes que formen el model de probabilitats

Posteriorment es va pensar en adjuntar un gràfic de les gaussianes a l'histograma, amb l'objectiu de proporcionar la oportunitat a l'usuari de veure si el programa estava definint els clústers correctament. Es va definir cada gaussiana mitjançant la mitjana i l'arrel quadrada de la covariància (sigma) que proporcionava la funció MixtureModel. Finalment l'alçada es va fer proporcional al pes de cada gaussiana per fer-ho de forma visual. Tot aquest procediment es va dur a terme mitjançant la funció *scipy.stats.norm.pdf.*

Cal destacar que, donat el fet que un histograma pels tres canals de color precisaria de quatre dimensions a la seva representació, aquesta opció només es va habilitar per la segmentació per barreja de gaussianes en imatges en escala de grisos.

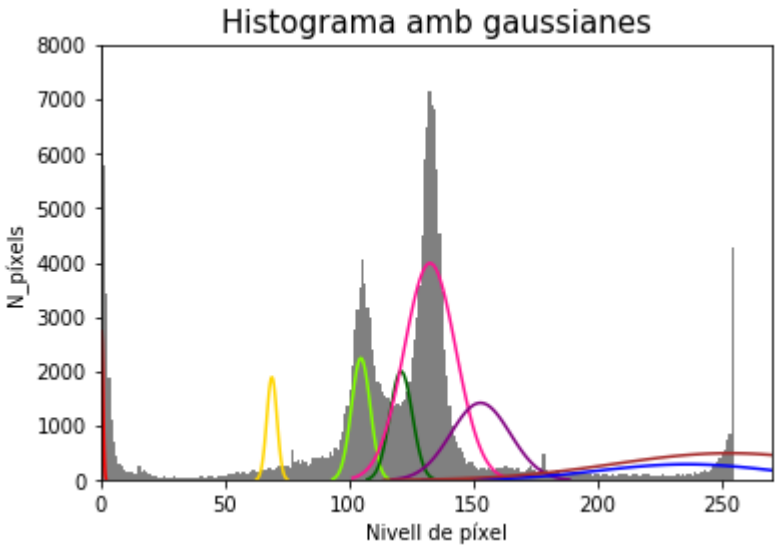

Figura 57: Resultat de la representació gràfica de l'histograma de la imatge amb les distribucions gaussianes trobades pel mètode de discriminació de gaussianes.

7. Manual d'ús

7.1. Manual d'usuari

- **Ajuda**

Seleccionar l'opció d'ajuda a la barra de menú superior. Un cop fet això té la opció de seleccionar entre ajuda sobre el copyright del programa, ajuda sobre els conceptes que es tracten i ajuda sobre la funcionalitat del mateix:

- Copyright: cliqui sobre la opció ***Qt*** i veurà una finestra emergent oficial de PyQt5 amb tota la informació
- Conceptes: cliqui sobre ***Conceptes*** i veurà una finestra emergent amb diferents títols en blau. Escolli el concepte sobre el que tingui dubtes i serà redirigit a una pàgina web oficial i contrastada on s'explicarà la base teòrica de cadascun. En trobareu: Python, PyQt5, Histograma, Contrast, segmentació i GMM.
- Funcionalitat: cliqui sobre ***L'aplicació*** i seleccioni quina de les funcions de l'aplicació li crea dubtes. Apareixerà una finestra emergent amb tota la informació que necessita saber. Entre ells trobarà: objectiu del programa, informació sobre els tipus de gaussian mixture model i quines condicions requereixen cadascun, com aplicar i esborrar el contrast, com s'obté la taula de característiques i com canviar de quina imatge ho fa, funcionalitat del botó vermell pan/(x,y) per arrossegar la imatge a la finestra i ampliar-la i veure les coordenades de píxel, com i on es guarden tant les captures de pantalla com les imatges processades i finalment què és i quines condicions precisa la representació d'un histograma amb gaussianes.

- **Canviar idioma**

Seleccionar a la barra de menú superior l'opció ***Idioma*** i escollir entre ***Català, Español, English***.

- **Capturar pantalla**

Per capturar una imatge de l'aplicació sencera en qualsevol moment seleccionar ***Fitxer > Captura de pantalla.*** La imatge es guardarà a la mateixa direcció on té l'aplicació amb el nom del dia i la hora en que s'ha fet. També hi ha l'opció de fer-ho des del teclat amb Ctrl+G.

- **Imprimir la imatge**

Si disposa d'una impressora pot imprimir a paper la pantalla de l'aplicació. Per fer això seleccioni ***Fitxer>Imprimir***, o mitjançant el teclat amb Ctrl+P. Se li obrirà una finestra de diàleg per configurar la impressora.

- **Mostrar una imatge**

Altre cop seleccionar ***Fitxer> Seleccionar imatge.*** Se'l redirigirà als fitxers del seu ordinador on podrà escollir una imatge a segmentar sempre que sigui de format *.png *.jpeg *.jpg *.bmp *.gif. També ho podrà fer mitjançant el teclat amb Ctrl+F.

- **Ampliar la imatge**

Per fer zoom in i zoom out sobre les imatges original i processada mantingui el cursor sobre la imatge a la zona que desitgi ampliar i mogui la rodeta del ratolí cap amunt per amunt per ampliar i avall per tornar enrere.

- **Arrossegar la imatge a la finestra i veure les coordenades de píxel**

Mitjançant el botó vermell ***pan/(x, y)*** es pot passar del mode arrossegar la imatge a veure les coordenades de píxel. Per canviar de mode cliqui sobre el botó

Si al passar el cursor sobre la imatge aquest te forma de mà, és indicatiu de que està en mode arrossegar imatge. En aquest cas simplement cliqui sobre la imatge i mogui el cursor per a que la imatge segueixi el moviment a la finestra.

Si, al contrari, el cursor té forma de fletxa, cliqui sobre el lloc exacte on vulgui saber les coordenades de píxel i aquestes apareixeran al quadre sobre la imatge al costat del botó vermell.

- **Mostrar histograma**

Aquesta opció s'habilitarà un cop carregui una imatge a processar. Seleccioni la opció ***Histograma>Histograma*** per a un histograma de la imatge original.

- **Exportar dades histograma**

Les dades del histograma es poden exportar clicant el botó dret del ratolí sobre aquest i aleshores ***Export... .***Un cop fet això podrà escollir entre exportar el plot de l'histograma en format imatge, finestra de Matplotlib o Scalalble Vector Graphics (SVG). També pot exportar les dades en format .csv. En cas de fer-ho com a imatge pot escollir el color de fons.

- **Aplicar contrast**

Seleccionar l'opció ***Contrast***, aleshores escollir entre els dos tipus de contrast disponibles; gamma i logarítmic.

- **Esborrar contrast**

Per esborrar el contrast clicar sobre la icona de la escombraria.

- **Segmentar**

Seleccionar ***Segmentació*** i després escollir entre les disponibles; Otsu, Watershed, Felzenszwalb, Quickshift, K-Means i gauxian mixture model (GMM).

Si es decideix per una segmentació K-Means n'hi ha de dos tipus; automàtica, on el número de clústers s'escull automàticament, i manual, on ha d'introduir el número de clústers desitjat. Aquesta xifra haurà de ser un enter en el rang [2,50], ambdós inclosos.

Si escull la gaussian mixture model també n'hi ha de dos tipus; per imatges en escala de grisos i per imatges a color. S'ha de tenir en compte que una imatge en color podrà ser segmentada amb els dos tipus de GMM, mentre que una imatge en grisos només podrà gaudir de la primera.

- **Guardar imatge processada**

Per guardar la imatge processada en qualsevol moment seleccioni sobre la barra de menú superior la opció ***Fitxer>Guardar imatge processada***, o mitjançant el teclat amb Ctrl+S.

- **Histograma amb gaussianes**

Només en el cas d'haver dut a terme una segmentació GMM en escala de grisos es podrà gaudir d'aquesta funció. Cliqui sobre la barra de menú superior a ***Histograma > Histograma amb gaussianes.*** Apareixerà l'histograma de la imatge original amb les gaussianes que ha trobat la funció de segmentació a sobre. Cada gaussiana d'un color i amb la una llegenda on trobarà la mitjana i la sigma de cadascuna.

- **Taula de propietats**

Per visualitzar la taula de propietats seleccionar a la barra de menú superior ***Propietats > Taula.*** Trobarà la etiqueta de cada zona, el número de píxels que formen l'àrea, el centroide en coordenades (x, y) i la intensitat de la zona.

- **Exportar dades de la taula en format .csv**

Per exportar les dades cliqui a la pantalla sobre el botó verd sota la taula que diu ***Guardar*** i seleccioni la carpeta de destí del fitxer.

- **Gràfic àrea/intensitat**

Per veure la finestra emergent que mostra el gràfic àrea/intensitat de cada zona etiquetada a la segmentació seleccionar a la barra de menú superior ***Gràfic.***

7.2. Manual de programador

En aquest apartat, tal com es va fer durant el primer període del projecte, es definirà què han de fer els futurs programadors per afegir processats, segmentacions, taules i gràfics.

- **Processament:**

Comencem afegint el processament a la classe Segmentada:

1) Inhabilitar histograma amb gaussianes.
2) Llegir imatge original i guardar a l'atribut ***imaori***.
3) Processar imatge i guardar-la a l'atribut ***process.***
4) Cridar a la funció toQImage() per a que es visualitzi.

Seguidament a la classe UI_TFG:

5) Crear una acció que es connecti amb el mètode que acabes de crear a la classe segmentada.
6) Crear un nou botó pel menú de processat a ***menuProcessament.***
7) Connectar el botó amb l'acció prèviament creada.
8) Afegir el títol en català, castellà i anglès del botó als mètodes ***retranslateUi(), idioma_castellano()*** i ***idioma_english()*** respectivament.

- **Segmentació:**

Primerament a classe Segmentada:

1) Inhabilitar histograma amb gaussianes.
2) Cridar al mètode pre_segm() per llegir la imatge corresponent.
3) Guardar la imatge segmentada a l'atribut ***segment.***
4) Cridar a la funció toQImage() per a que es visualitzi.

Seguidament es repeteixen els passos 5-8 però al menú ***menuSegmentaci***

- **Taula:**

Començant a la classe Segmentada:

1) Afegir la obtenció de la propietat que t'interessa.
2) Afegir-la a la taula tot convertint el valor numèric a string.

- **Gràfic**

1) Crear un nou plot amb eines de Matplotlib a la classe Segmentada
2) Seguir els passos 5-8 de processament però amb el menú ***menuGr_fic.***

8. Impacte mediambiental i social

Qualsevol acció humana té un impacte al medi ambient i a la societat, és per això que, per qüestions ètiques, una part necessària de tot projecte d'enginyeria és l'anàlisi dels efectes de les accions que hem realitzat amb la finalitat de dur-lo a terme, per tal de comparar si és recomanable seguir endavant.

8.1. Impacte mediambiental

Les influències mediambientals negatives han sigut escasses durant aquest projecte. El desenvolupament del software ha implicat com a única eina de treball un ordinador que ja s'havia adquirit amb anterioritat per altres propòsits. També s'ha tingut en compte l'impacte mediambiental de l'ús del transport per a les reunions periòdiques i el consum d'energia de l'ordinador.

8.1.1. Reciclatge d'aparells elèctric i electrònics

La majoria de les deixalles electròniques s'envien a abocadors d'escombraries o s'incineren, creant un gran impacte negatiu al medi ambient per l'alliberament de materials com el plom, el mercuri o el cadmi al terra, les aigües subterrànies i l'atmosfera. No obstant, molts dels materials utilitzats a la fabricació d'equips informàtics poden ser recuperats per un posterior ús en noves fabricacions. En són exemples l'estany, el silici, el ferro i l'alumini i una varietat de plàstics.

És per això que la Unió Europea va implementar un sistema de reciclatge de residus electrònics i elèctrics al febrer de 2003, sota la directiva d'Equips (RAEE Directiva 2002/96 / CE), inspirada en un sistema similar que s'havia imposat a Suïssa a l'any 1991.

Darrerament, la normativa ha sigut actualitzada a la Directiva 2011/65 / UE del Parlament Europeu i de Consell , de 8 de juny de 2011 i a la directiva 2012/19 / UE del Parlament Europeu i de Consell, de 4 de juliol de 2012 [68].

En el cas d'Espanya en concret, es va produir la transposició de les dues normatives europees que s'acaben de mencionar [69]:

- **La Directiva 2011/65 / UE del Parlament Europeu i de Consell, de 8 de juny de 2011:** orientada a la prevenció. Aquesta directiva contenia restriccions en relació a la utilització de determinades substàncies perilloses en aparells elèctrics i electrònics. Es va traslladar a l'ordenament jurídic espanyol mitjançant el Reial Decret 219/2013, de 22 de març, sobre restriccions a la utilització de determinades substàncies perilloses en aparells elèctrics i electrònics.
- **La Directiva 2012/19 / UE del Parlament Europeu i de Consell, de 4 de juliol de 2012**: orientada a la gestió dels RAEE (residus d'aparells electrònics i elecèctrics). Aquesta directiva

contemplava les restriccions al tractament de residus d'aparells elèctrics i electrònics. Es va incorporar a la normativa espanyola mitjançant el Reial Decret 110/2015, de 20 de febrer, sobre residus d'aparells elèctrics i electrònics. Segons aquest, com a mínim es retiraran els següents components i substàncies:

- Condensadors que continguin policlorobifenils (PCB)
- Components o RAEE que continguin mercuri
- Piles i acumuladors.
- Targetes de circuits impresos
- Cartutxos de tòner, de líquid i pasta.
- Plàstics que continguin materials piroretardants bromats.
- Residus d'amiant
- Tubs de raigs catòdics.
- Clorofluorocarburs (CFC), (HCFC), hidrofluorocarburs (HFC), hidrocarburs (HC) i amoníac (NH3).
- Llums de descàrrega de gas.
- Pantalles de cristall líquid (juntament amb la seva carcassa si escau)
- Components que continguin fibres ceràmiques refractàries

8.1.2. Impacte mediambiental del transport

També es comptabilitzen els viatges necessaris per les reunions, ja que no consten només de moviments en transport públic sinó que també impliquen un medi de transport privat. El cotxe responsable d'aquests trajectes és un Ford K de l'any 2011 amb benzina com a combustible, qualificat com a turisme lleuger amb un nivell d'emissions EURO 5/V de classe C per la DGT [70].

8.2. Impacte social

En contraposició amb l'apartat anterior, l'impacte social del nostre projecte és més que positiu. Al ser un projecte d'enginyeria biomèdica, està especialment dissenyat per fer un servei a la comunitat, ajudant als sanitaris de tot el món a diagnosticar malalties amb facilitat i així augmentar la fluïdesa dels serveis hospitalaris.

Gràcies al nostre projecte, tenint en compte una probable continuació del mateix on es segueixin els suggeriments proposats, es podrà fer un anàlisi exhaustiu de la simptomatologia dels pacients de forma automàtica, alliberant esforç i temps al personal sanitari alhora que s'incrementa la minuciositat en el mateix. Esperem que el projecte serveixi d'inspiració per altres enginyers que vulguin col·laborar a una causa tan lloable.

En conclusió, es creu que l'equilibri entre els efectes negatius i positius del projecte està balancejat i per tant és recomanable posar-lo en pràctica.

9. Conclusions

El treball realitzat en aquesta tesis ha sigut extens. La memòria només recull la part del treball que ha suposat una aportació encertada per l'aplicació, però es van explorar prèviament altres camins fins a arribar a aquest punt, en especial en relació a la implementació de noves segmentacions.

El projecte no només ha precisat d'un llarg procés d'aprenentatge, valoració i proves a nivell de processat d'imatge sinó que ha sigut també necessari l'estudi autodidàctic de llenguatges orientats a objectes per a l'actualització del disseny i codificació de la interfície d'usuari. S'ha requerit d'una gran inversió de temps per a obtenir les habilitats necessàries per una adient manipulació d'aquest.

Cal destacar que el fet que aquesta tesis hagi sigut la continuació d'una altra ha comportat l'esforç suplementari de comprendre un projecte sencer realitzat per un professional aliè, procés especialment difícil tractant-se d'una aplicació amb un alt grau de programació informàtica.

De la mateixa manera que s'han definit els objectius, s'han considerat les conclusions segons la seva relació amb el software o la interfície d'usuari.

9.1. Actualitzacions del software

En aquest cas, es considera que l'aplicació ha triomfat en la implementació dels nous mètodes de segmentació proposats, que han permès una segmentació ràpida i precisa. En concret l'aprofundiment en la segmentació per Barreja de Gaussianes ha aportat una diferència significativa en comparació amb les antigues metodologies. També s'ha habilitat una opció per a que l'usuari tingui la capacitat de marcar alguns dels paràmetres, com és el cas de la segmentació K-Means manual.

Per una altra banda, en referència a la eficiència del codi utilitzat, es conclou que totes les funcions han estat optimitzades i que s'ha simplificat el codi, tot evitant redundàncies. N'és un exemple la creació de funcions que resumeixen processos repetitius, com per exemple el procés de lectura de la imatge (mètode pre_segm), el canvi automàtic del format de la mateixa (mètode toQImage), així com la construcció de la taula de característiques, que ja no implica la repetició de la segmentació a la imatge cada cop que es crida.

També s'han afegit eines complementàries. Per començar, s'ha atorgat al programa tot tipus de mètodes per a guardar la informació i els resultats obtinguts: captures de pantalla de l'aplicació i de la imatge processada, exportació de dades de l'histograma i de la taula de característiques i enviament del gràfic de l'histograma al ordinador privat de l'usuari en diferents formats. També s'ha implementat la capacitat de visualitzar l'histograma de la imatge d'entrada, la possibilitat d'eliminar el contrast

aplicat en cas que l'usuari decideixi segmentar sense aquest, l'enllaç directe a una impressió en paper dels resultats i, per últim, l'accessibilitat a informació tant de restriccions a l'hora d'utilitzar l'aplicació com instruccions per a procedir a un correcte processat de la imatge.

Cal afegir que en tot cas s'ha tingut en compte la variabilitat entre els diferent tipus d'imatge biomèdica, tractant de no limitar la utilització de l'aplicació a un camp específic.

9.2. Actualitzacions de la GUI

S'ha aconseguit dirigir l'aplicació a un ventall més ampli d'usuaris fent-la disponible en diferents llengües entre les que trobem el català, el castellà i l'anglès. A més, el codi ha deixat accessible la possibilitat d'afegir nous idiomes si es precisa.

Mitjançant el disseny gràfic de la pantalla principal s'ha apel·lat en un sentit més directe als usuaris que s'han considerat potencials, els professionals de la medicina. A més, aquesta ha adquirit un semblant més atraient. En aquest sentit, els widgets també han sigut modificats dotant-los d'un aire més senzill i familiar per a l'usuari a través de la configuració de diferents tonalitats per cadascun i l'acompanyament de cada funció amb una icona.

La qualitat de la visualització de les imatges s'ha vist incrementada gràcies a la utilització d'eines pròpies de Qt (QGraphicsView) i el canvi de format de la imatge (QImage).

Per últim, s'ha prestat especial atenció a la orientació de l'usuari dins l'aplicació mitjançant finestres emergents amb instruccions i la incorporació d'un ampli apartat d'ajuda. En concret, s'han creat una sèrie de *warnings* que avisen a l'usuari quan s'està fent un ús erroni del programa, tot informant-lo de les passes a seguir a continuació. N'és un exemple la finestra emergent que apareix al intentar segmentar la imatge amb el mètode K-Means manual en cas que l'usuari seleccioni un valor de número de clústers fora del rang que marca la lògica. També s'ha creat un menú d'ajuda on l'usuari pot trobar tant ajuda sobre el funcionament i la metodologia de l'aplicació com dels conceptes que en ella es tracten. En aquest segon cas s'han afegit enllaços directes a pàgines webs contrastades on s'expliquen amb detall.

Totes aquestes actualitzacions ratifiquen que els mètodes proposats en el marc d'aquesta tesi són vàlids i demostren la seva aplicabilitat i eficiència en l'establiment d'una relació més directa entre el professional mèdic i l'àmbit del processat d'imatges que ajudarà a perfilar i polir els processos de segmentació i reconeixement d'objectes en imatges mèdiques. Sens dubte, la metodologia dissenyada, implementada i validada en aquesta tesi pot ser estesa a la segmentació de tot tipus d'imatges d'àmbit

biomèdic. És per aquest motiu que es considera que s'han consumat tots els objectius proposats per aquest projecte.

10. Continuació del treball

Durant aquesta segona etapa del programa s'han perfeccionat tècniques de segmentació, s'ha afegit la possibilitat de visualització de diferents tipus d'histograma, s'ha canviat el disseny gràfic per fer-lo més atraient al públic objectiu, s'ha afegit un apartat d'ajuda intensiva als usuaris menys familiaritzats amb els conceptes i, per últim, s'ha optimitzat el codi per reduir el temps i la càrrega d'execució.

No obstant això, i tot i haver sigut la segona part d'un projecte que va dur a terme una companya, s'han establert una sèrie de tècniques que encara es poden implementar per fer-lo més complet.

1. **Focalització en un tipus específic d'imatge mèdica:** durant la complementació de tècniques de segmentació es van haver de prendre decisions tenint en compte que l'aplicació havia de respondre a qualsevol tipus d'imatge mèdica, sempre en detriment de la qualitat d'aquesta. És per això que la especialització en un tipus d'imatge mèdica, com pot ser una fMRI, és clau per estretir el ventall de condicions de les segmentacions i focalitzar-se en zones de la imatge amb una forma o mida concreta.

2. **Recerca d'altres tècniques de segmentació:** després d'haver avaluat les diferents tècniques de segmentació precises, es considera apropiada la comparació de resultats amb tècniques difuses, en especial Fuzzy C-Means, ja que varis estudis proven la seva eficiència [71] [29].

3. **Visualització de la etiqueta d'un clúster en resposta a moviments de ratolí:** l'objectiu és permetre a l'usuari seleccionar una regió de la imatge i mostrar a quina etiqueta pertany la selecció i les propietats extretes d'aquesta. [1]

4. **Compatibilitzar el programa amb imatge DICOM utilitzades en els hospitals.** [1]

5. **Habilitació de processat d'imatges 3D:** les imatges 3D en la medicina formen un camp emergent en el món actual. És per això que el processat d'imatges en tres dimensions permetria una millor actualització i adaptació a les necessitats de diagnòstic per imatges d'avui en dia.

6. **Processament de varies imatges simultànies, amb els seus corresponents gràfics i taula de dades:** l'objectiu seria establir comparacions entre diferents imatges d'una mateixa mostra per ajudar al reconeixement d'objectes.

11. Gestió econòmica

De la mateixa manera que a la primera part del projecte, s'estimaran amb la major precisió possible els costos del projecte tot desglossant aquests en cinc seccions diferents; pressupost de recursos humans, pressupost del hardware, pressupost de software, costos indirectes i costos per imprevistos.

11.1. Pressupost de Recursos Humans

Aquest projecte s'ha dut a terme en dues fases per dues enginyeres diferents. No obstant això, s'ha tingut en compte la contractació d'una sola enginyera per fer tota la feina, ja que la necessitat de dues persones ha estat causada per la disponibilitat de cadascuna, no pe runa millora en l'eficiència. Així, es sumaran les hores que es van dedicar a la primera part del projecte per comptabilitzar l'esforç total.

Segons El diari dels Enginyers Industrials de Catalunya (COEIC/AEIC, 2017) el salari mitja d'un enginyer jove sense experiència seria d'uns 22.000 € a l'any per jornada completa de 8 h.

Rol	Durada (mesos)	Salari (€)	Cost total (€)
Enginyera sense experiència	10	1.833,33	18.333,3
Total			18.333,3

Taula 5: Pressupost destinat a la contractació del personal per dur a terme el projecte

11.2. Pressupost del Hardware

En aquest apartat es farà un càlcul aproximat del cost d'utilització del hardware tot tenint en compte les hores que s'ha utilitzat l'ordinador per fer el projecte, la vida útil de la màquina i l'amortització per hora. Si durant un any laboral es treballen 2.000 hores amb un contracte de jornada completa, en els 10 mesos que dura el nostre projecte s'hauran invertit 1.667 hores.

Calculem doncs l'amortització, que s'ha considerat el preu del producte entre els anys de vida útil i les hores d'utilització per al projecte:

$$Amortització = \frac{Preu}{Anys\ de\ vida\ útil \cdot Hores\ laborables/any} = \frac{499€}{5\ anys \cdot 2000\ hores\ /any} = 0{,}0499€/h$$

Equació 4: Càlcul de l'amortització segons el procediment de [1]

El cost total doncs, són les hores que s'ha utilitzat el producte per l'amortització del mateix:

Producte	Preu (€)	Vida útil (Anys)	Amortització (€/h)	Hores d'ús (h)	Cost total (€)
Portàtil HP 15-db0096ns	499	5	0,0499	1.667	83,18
Total					83,18

Taula 6: Pressupost destinat a la utilització d'un ordinador per desenvolupar el projecte

11.3. Pressupost del Software

Tots els softwares utilitzats durant el projecte són de programari lliure, amb excepció del Microsoft Office. La llicència d'aquest provoca una despesa de 99,99€ a l'any.

Producte	Preu (€)
Microsoft Office	99,99
Total	99,99

Taula 7: Pressupost destinat a la utilització de softwares de pagament.

11.4. Costos indirectes

En aquest apartat s'inclouran totes aquelles despeses necessàries per dur a terme el projecte però que no es necessitin explícitament per a la creació del software. És a dir, es tindrà en compte la factura de la electricitat, fibra òptica, despatx de treball, transport a les reunions amb el director del projecte i una forma de mantenir-se en contacte constant amb el mateix, com pot ser una línia mòbil amb capacitat de missatgeria instantània.

11.4.1. Electricitat

Per poder estimar el preu de la electricitat s'ha calculat el consum energètic de l'ordinador HP 15-db0096ns, el preu de l'electricitat en el seu espectre d'horaris a la companyia Endesa i el número d'hores d'utilització dins l'horari laboral. Aquest últim s'ha considerat com el 95% del total d'hores, tenint en compte descansos i períodes de reflexió. Així, el consum elèctric de l'ordinador és de 65Wh

[72], el preu de la electricitat tenint en compte un horari de 8:00AM – 17:00PM, amb una hora per dinar, és de 0,1387€/kWh de mitja [73], i finalment les hores d'utilització són 1583,65h.

El càlcul total de la energia és el següent:

$$C_{Electricitat} = consum\ ordinador * preu\ electricitat * hores\ d'utilització =$$

$$= 0{,}065\ kWh \cdot 0{,}1387 \frac{€}{kWh} \cdot 1583{,}65h = 14{,}27\ €$$

Equació 5: Càlcul del cost de la electricitat.

11.4.2. Fibra òptica i telefonia mòbil

Després de comparar entre diferents companyies, s'ha considerat un contracte amb O_2 , ja que disposa de la xarxa de fibra i cobertura 4G de Movistar a un preu més econòmic. Aquesta tarifa inclou fibra òptica simètrica de 600Mb, línia fixe amb trucades il·limitades a fixes i mòbils nacionals i una línia mòbil de 25GB amb trucades il·limitades. La quota de línia està inclosa al preu, la instal·lació és gratuïta i el contracte manca de permanència. El preu final del producte és de 50€/mes [74].

El càlcul total de la despesa d'Internet i telefonia mòbil és el següent:

$$C_{Internet} = cost\ mensual\ tarifa * durada\ projecte = \frac{50€}{mes} \cdot 10\ mesos = 500€$$

Equació 6: Càlcul del cost d'internet i telefonia mòbil.

11.4.3. Espai de treball

Pel càlcul del lloguer del despatx on treballar s'ha considerat el preu del lloguer del pis de residència de l'autor del projecte, tenint en compte el número d'hores que es dediquen a la feina en un contracte de jornada completa. Així, la despesa total és:

$$\frac{C_{lloguer}}{h} = \frac{preu\ lloguer\ mensual}{hores\ d'utilització\ del\ domicili\ destinades\ al\ projecte} =$$

$$= \frac{850\ €/mes}{160\ h/mes} = 5{,}31 \frac{€}{h}$$

Equació 7: Càlcul del cost del lloguer per cada hora d'utilització.

$$C_{lloguer} = \frac{C_{lloguer}}{h} * durada\ projecte = \ 5{,}31 \frac{€}{h} \cdot 1667\ h = 8851{,}77\ €$$

Equació 8: Càlcul del cost del lloguer per la durada total del projecte.

11.4.4. Transport

El transport es realitza un cop per setmana des del lloc de residència (Sant Pere de Vilamajor) fins a la facultat EEBE, a Barcelona. Es precisa de transport privat des de casa fins a la estació de tren més propera a Llinars del Vallès, i transport públic entre les zones de Llinars i Barcelona. El cotxe de l'usuari és un model Ford K que consumeix 6 litres als 100 km, i la qualitat de bitllet de tren és la de T10 de 3 zones. Sense tenir en compte el preu del cotxe de l'usuari, el càlcul total de la despesa és:

$$C_{trans_{privat}} = consum\ cotxe * \frac{distància}{trajecte} * preu\ benzina * N\frac{viatges}{mes} * durada\ projecte =$$

$$= \frac{6L}{100km} \cdot 5\ km \cdot 1{,}3\frac{€}{L} \cdot 8\frac{viatges}{mes} \cdot 10\ mesos = \ 31{,}2€$$

Equació 9: Càlcul de la despesa en transport privat.

$$C_{trans_{públic}} = \frac{preu\ tiquet}{10\ viatges} * \ N\frac{viatges}{mes} * durada\ projecte =$$

$$= \frac{30€}{10\ viatges} \cdot 8\frac{viatges}{mes} \cdot 10\ mesos = 240€$$

Equació 10: Càlcul de la despesa en transport públic.

$$C_{transport} = C_{trans_{privat}} + \ C_{trans_{públic}} = 31{,}2€ + 240€ = 271{,}2€$$

Equació 11: Càlcul de la despesa total de transport per atendre a les reunions durant el projecte.

11.4.5. Imprevistos

En tot projecte s'ha de tenir en compte possibles imprevistos, els quals acaben resultant sempre en un endarreriment en la finalització del mateix. Aquest temps extra repercuteix en les despeses que s'han calculat anteriorment, ja que depenen directament de la durada del projecte, amb excepció del software, ja que és de pagament anual i no es calcula un retard tan elevat. És per això que considerarem un possible endarreriment de com a màxim 1 mes i mig, ja que representa un 15% del temps estimat inicial, fent que el total d'hores treballades sigui de 1917h.

Com a conseqüència, els nous costos costs calculats amb les equacions corresponents a cadascun quedarien:

$$C_{personal} = 21.083{,}295\ €$$

$$C_{hardware} = 95{,}66€$$

$$C_{software} = 99{,}99€$$

$$C_{electricitat} = 16{,}41€$$

$$C_{internet} = 575€$$

$$C_{lloguer} = 10.179{,}27€$$

$$C_{transport_{privat}} = 35{,}88€$$

$$C_{transport_{públic}} = 300€$$

11.5. Total

La suma de totes les despeses durant tot el temps que dura el projecte dona un total de:

ROL	Preu (€)
Recursos humans	21.083,295
Hardware	95,66
Software	99,99
Indirectes	$16{,}41 + 575 + 10.179{,}27 + 35{,}88 + 300 = 11.106{,}56$
Total	**32.385,5€**

Taula 8: cost total de tot el projecte amb la desviació per imprevistos aplicada.

Bibliografia

[1] S. Bernaus, «Plataforma per la visualització i anàlisi d'imatges d'anatomia patològica,» Universitat Politècnica de Catalunya, Barcelona, 2019.

[2] A. Bereciartua Pérez, «Desarrollo de algoritmos de procesamiento de imagen avanzado para interpretación de imágenes médicas. Aplicación a segmentación de hígado sobre imágenes de Resonancia Magnética multisecuencia.,» Universidad de País Vasco (UPV/EHU), Bilbao, 2016.

[3] A. Restrepo, «Procesamiento de imágenes,» 1999.

[4] N. Holtz y W. Rasdorf, «An evaluation of programming languages and language features for engineering software development.,» *Eng. Comput.,* vol. 3, nº 4, pp. 183-199, 1988.

[5] B. Stroustrup, «What is object-oriented programming?,» *IEEE Softw. ,* vol. 5, nº 3, pp. 10-20, 1988.

[6] B. Cox, Object Oriented Programming., Reading: Addison-Weskey, 1986.

[7] G. L. Fenves, «Object-oriented programming for engineering software development,» *Engineering with Computers,* vol. 6, pp. 1-15, 1990.

[8] L. Prechelt, «An empirical comparison of seven programming languages,» *Computer,* vol. 33, nº 10, pp. 23-29, 2000.

[9] S. Cass, *The 2018 Top Programming Languages,* IEEE Spectrum, 2018.

[10] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu i t. s.-i. contributors, «scikit-image: image processing in Python,» *PeerJ, Life and environment,* vol. 2, 2014.

[11] N. D. Draper, User Center System Design: New Perspectives on Human-Computer Interaction, Hillsdale, N.J: SW, editors, 1986.

[12] J.-W. Chen i J. Zhang, «Comparing Text-based and Graphic User Interfaces for Novice and Expert Users,» *AMIA Annu Symp Proc. ,* p. 125–129, 2007.

[13] B. Shneiderman, Designing The user interface, Strategies fo effective Human computer interaction, Addison-Ewsley, 1998.

[14] J. Royo, Diseño Digital, Ediciones Paidós Ibérica, 2018.

[15] E. Hutchins, J. Hollan i D. Norman, «Direct manipulation interfaces.,» *Human-Computer interaction,* vol. 1, p. 311–338, 1985.

[16] I. E. Sutherland, «Sketchpad: A man-machine graphical communication system.,» *Proc of the Spring Joint Comp Conf,* vol. 2, núm. 5, p. 329–346, 1963.

[17] L. H. Boyd, «The Graphical User Interface Crisis: Danger and Opportunity.,» *Eric,* p. 23, 1990.

[18] R. M, «An empirical comparison of menu-selection (CUI) and desktop (GUI) computer programs carried out by beginners and experts.,» *Behaviour & Information Technology,* vol. 11, p. 227–36, 1992.

[19] D. S i B. R, «An experimental investigation of the roles of the computer interface and individual characteristics in the learning of computer systems.,» *Int J Human-Computer Interaction.,* vol. 4, núm. 2, p. 143–72, 1992.

[20] Temple, Barker & Sloane, Inc ., «The Benefits of the Graphical User Interface: A Report on New Primary Research Redmond.,» *Wash: Microsoft Corp,* 1990.

[21] S. N i K. D, «Comparing response time, errors, and satisfaction between text-based and graphical user interfaces during nursing order tasks.,» *J Am Med Inform Assoc.,* vol. 7, pp. 164-176, 2000.

[22] P. Patidar, S. Srivastava, M. Gupta i A. K. Nagawat, «Image De-noising by Various Filters for Different Noise,» *International Journal of Computer Applications,* vol. 9, núm. 4, pp. 45-50, 2010.

[23] C. Pinilla, A. Alcalá i J. Ariza, «Filtrado de imágenes en el dominio de la frecuencia,» *Revista de la Asociación Española de Teledetección,* vol. 8, núm. 8, pp. 1-5, 1997.

[24] L. M. Quintana Vivanco, F. Hubert Sánchez i C. E. Marañón, «Optimización de los filtros Mediana - Gaussiano,» de *XV convención y feria internacional*, Habana, 2013.

[25] J. Angulo i J. Serra, «Segmentación de Imágenes en Color utilizando Histogramas Bi-Variables,» *Computación y Sistemas,* vol. 8, núm. 4, pp. 303-316, 2005.

[26] D. M. Hawkins, «The Problem of Overfitting,» *J. Chem. Inf. Comput. Sci.,* vol. 44, pp. 1-12, 2004.

[27] A. Rabelo, «Machine Learning: ¿qué es y cuál es su influencia en el marketing digital?,» 21 11 2018. [En línia]. Available: https://rockcontent.com/es/blog/machine-learning/. [Últim accés: 15 05 2020].

[28] N.Gordillo, E.Montseny i P.Sobrevilla, «State of the Art Survey on MRI Brain Tumor Segmentation,» 2013.

[29] B. Prados Suárez, «Desarrollo de Modelos para la Segmentación Difusa de Imágenes a Color,» Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad de Granada E.T.S Ingeniería informática, Granada, 2006.

[30] J. Batlle, A. Casals, J. Freixenet i J. Martí, «A review on stra- tegies for recognizing natural objects in colour images of outdoor scenes.,» *Image and Vision Computing,* vol. 18, núm. 6-7, pp. 515-530, 2000.

[31] T. Q. Chen i Y. Murphey, «Color image segmentation - An innovative approach,» *Pattern Recognition,* vol. 35, núm. 2, pp. 395-405, 2002.

[32] A. Moghaddamzadeh i N. Bourbakis., «A fuzzy region growing approach for segmentation of color images.,» *Pattern Recognition,* vol. 30, núm. 6, pp. 867-881, 1997.

[33] L. Lucchese i S. -K. Mitra, «Colour image segmentation: a state-of-the-art survey.,» *Proceedings of the Indian Nacional Science Academy,* vol. 67, núm. 2, pp. 21-207, 2001.

[34] H. D. Cheng i Y. Sun, «A hierarchical approach to color image segmentation using homogenity.,» *Image Processing,* vol. 9, núm. 12, pp. 2071-2082, 2000.

[35] H. D. Cheng i J. Li, «Fuzzy homogeneity and scale-space approach to color image segmentation.,» *Pattern Recognition,* vol. 36, núm. 7, pp. 1545-1562, 2003.

[36] J. Fan, D. K. Y. Yau, A. K. Elmagarmid i W. G. Aref, «Automatic image segmentation by integrating color-edge extractionand seeded region growing.,» *Image Processing,* vol. 10, núm. 10, pp. 1454-1466, 2001.

[37] P. K. Sahoo, S. Soltani i A. K. C. Wong, «Survey of thresholding techniques,» *Computer Vision, Graphics and Image Processing,* vol. 41, núm. 2, pp. 233-260, 1988.

[38] M. Lather i D. P. Singh, «Conferència internacional sobre la intel·ligència computacional i la ciència de dades,» 2020.

[39] S. Agarwal, S. Madasu, M. Hanmandlu i S. Vasikarla, «A comparison of some clustering techinques via color segmentation,» de *International Conference on Information Technology: Coding and Computing*, 2005.

[40] G. Dong i M. Xie, «Color clustering and learning for image segmentation based on neural networks,» *Neural Networks,* vol. 16, núm. 4, pp. 925-936, 2005.

[41] D. Xu i Y. Tian, «A Comprehensive Survey of Clustering Algorithms,» *Annals of data science,* vol. 2, p. 165–193, 2015.

[42] M. Egmont-Petersen, D. d. Ridder i H. Handels, «Image processing with neural networks - a review,» *Pattern Recognition,* vol. 35, núm. 10, pp. 2279-2301, 2002.

[43] L. S. Davis., «A survey of edge detection techniques.,» *Computer Graphics and Image Processing,* vol. 4, pp. 170-248, 1975.

[44] C. Russ, «The Image Processing Handbook,» *CRC Press,* 1994.

[45] H. D. Cheng, X. H. Jiang, Y. Sun i J. Wang, «Color image segmentation: advances and propspects,» *Pattern Recognition,* vol. 34, núm. 12, pp. 2259-2281, 2001.

[46] R. Adams i L. Bischof, «Seeded region growing.,» *Pattern Analysis and Machine Intelligence.,* vol. 16, núm. 6, pp. 641-647, 1994.

[47] C.-H. Chuang i W.-N. Lie, «Region growing based on extended gradient vector flow fiels modeflo multiple objects segmentation,» de *International Conference on Image Processing*, 2001.

[48] J. Garcia-Consuegra, G. Cisneros i E. Navarro, «A sequential echo algorithm based on the integration of clusteingand region growing techniques,» de *Geoscience and Remote Sensing Symposium*, 2000, pp. 648-650.

[49] M. C. D'Ornellas, «A multi-scale gradient approach for color-based morphological segmentation,» de *15th International Conference on Pattern Recognition*, 2000.

[50] G. Kukielka i J. Woznicki, «Hierarchical Method Digital Image Segmentation Using Multidimensional Mathemtical Morphology,» vol. 2124, pp. 581-, 2001.

[51] A. Altameem, «Medical Iage Segmentation Methods,» *Algorithms and Applications,* 2014.

[52] N. Ikonomakis, K. N. Pataniotis i A. N. Venetsanopoulos, «Unsupervised seed determination for region-based clor image segmentation scheme.,» de *International Conference on Image Processing*, 2000.

[53] L. Griffin, A. Colchester i G. Robinson, «Scale and segmenttion of grey-level images using maximum gradient paths,» *Image and Vision Computing,* vol. 10, núm. 6, pp. 389-402, 1992.

[54] K. P. Murphy, «Mixture models and the EM algorithm,» de *Machine Learning: a Probabilistic Perspective*, London, The MIT Press, 2011, pp. 337-381.

[55] «Cálculo de probabilidades,» de *Distribuciones de probabilidad*, Sergas, 2014, pp. 3-5.

[56] F.-A. H, G. MM, R. JL y S. JA, Cálculo de probabilidades y, Barcelona: Editorial Ariel, 1994.

[57] Scikit-Learn developers, «Gaussian Mixture Models,» Python, 2019. [En línia]. Available: https://scikit-learn.org/stable/modules/mixture.html#gmm. [Últim accés: 18 05 2020].

[58] PennState, College of Health and Human Development, «AIC vs BIC,» [En línia]. Available: https://www.methodology.psu.edu/publications/. [Últim accés: 18 05 2020].

[59] K. P. Murphy, «EM for GMMs,» de *Machine Learning, A Probabilistic Perspective*, London, The MIT Press, 2012, pp. 381-383.

[60] M. Krasser, «Latent variable models - part 1: Gaussian mixture models and the EM algorithm,» *GitHub,* 2019.

[61] I. Myung, «Tutorial on maximum likelihood estimation,» *Journal of Mathematical Psychology,* pp. 90-100, 2003.

[62] T. Ortíz, «Variables latentes y algoritmo EM,» 09 2015. [En línia]. Available: https://tereom.github.io/est-multivariada-15/09-EM.html. [Últim accés: 18 05 2020].

[63] J. d. A. Madeja, «Guía para la redacción de casos de uso,» 2017. [En línia]. Available: http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/416.

[64] scikit-image development team, «scikit-image, image processing in Python,» 10 2019. [En línia]. Available: https://scikit-image.org/. [Últim accés: 06 05 2020].

[65] L. Campagnola, «Pyqtgraph documentation,» 2011. [En línia]. Available: http://www.pyqtgraph.org/documentation/introduction.html. [Últim accés: 14 05 2020].

[66] S. S. Varshney, N. Rajpal i R. Purwar, «Comparative Study of Image Segmentation Techniques and Object Maching Using Segmentation,» 2009.

[67] R. Dash, R. L. Paramguru i R. Dash, «Comparative Analysis of Supervised and Unsupervised Discretization Techniques,» *International Journal of Advances in Science and Technology,* vol. 2, núm. 3, pp. 29-37, 2011.

[68] U. Baid, S. Talbar i S. Talbar, «Comparative Study of K-means, Gaussian Mixture Model, Fuzzy,» 2017.

[69] «¿Cuánto gasta un aparato eléctrico? ¿Cuánta energía consume?,» 2010-2018. [En línia]. Available: http://www.electrocalculator.com. [Últim accés: 14 05 2020].

[70] G. Lahera, «Selectra,» 15 04 2020. [En línia]. Available: http://www.selectra.es. [Últim accés: 30 04 2020].

[71] Telefónica, «Tarifa Fibra 600Mb y Móvil 25GB,» O2, 2020. [En línia]. Available: http://o2online.es/fibra-y-movil/. [Últim accés: 30 04 2020].

[72] European Comission, «Waste Electrical & Electronic Equipment (WEEE),» 23 03 2020. [En línia]. Available: https://ec.europa.eu/environment/waste/weee/index_en.htm. [Últim accés: 10 05 2020].

[73] Ministerio de Agricultura, Alimentación y Medio Ambiente, *Real Decreto 110/2015,* España, 2015.

[74] A. d. Barcelona, «El distintivo ambiental de la DGT,» 2020. [En línia]. Available: https://ajuntament.barcelona.cat/qualitataire/es/afectacions-la-mobilitat/el-distintivo-ambiental-de-la-dgt. [Últim accés: 14 05 2020].

[75] A. Mustaqeem, A. Javed i T. Fatima, «An Efficient Brain Tumor Detection Algorithm Using Watershed and Tresholding Based Segmentation,» 2012.

[76] Departamento de Informática, Universidad de Valladolid, «Casos de Uso,» 5 de julio de 2016.

[77] A. Kornilov i I. Safonov, «An Overview of Watershed Algorithm Implementations in Open Source Libraries,» *Journal of Imaging,* vol. 4, 2018.

[78] Ekhumoro, «Stack overflow,» 15 10 2019. [En línia]. Available: https://stackoverflow.com/questions/35508711/how-to-enable-pan-and-zoom-in-a-qgraphicsview. [Últim accés: 20 02 2020].

Annex A

A1. Script photoviewer

```
from PyQt5.QtCore import Qt

from PyQt5.QtGui import QImage, QPixmap, QPalette, QPainter, QScreen

from PyQt5.QtPrintSupport import QPrintDialog, QPrinter

from PyQt5.QtWidgets import QLabel, QSizePolicy, QScrollArea, QMessageBox, QMainWindow, QMenu, QAction, \
    qApp, QFileDialog, QApplication, QTableWidget, QTableWidgetItem

import pyautogui

from datetime import datetime

from PyQt5 import QtCore, QtGui, QtWidgets

from segmentada import Segmentada

import matplotlib.image as mpimg

import pyqtgraph as pg

from pyqtgraph import PlotWidget, plot


class PhotoViewer(QtWidgets.QGraphicsView):
    photoClicked = QtCore.pyqtSignal(QtCore.QPoint)


    def __init__(self, parent):
        super(PhotoViewer, self).__init__(parent)
        self._zoom = 0
        self._empty = True
        self._scene = QtWidgets.QGraphicsScene(self)
```

```
    self._photo = QtWidgets.QGraphicsPixmapItem()
    self._scene.addItem(self._photo)
    self.setScene(self._scene)
    self.setTransformationAnchor(QtWidgets.QGraphicsView.AnchorUnderMouse)
    self.setResizeAnchor(QtWidgets.QGraphicsView.AnchorUnderMouse)
    self.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOff)
    self.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOff)
    self.setBackgroundBrush(QtGui.QBrush(Qt.transparent))
    self.setFrameShape(QtWidgets.QFrame.NoFrame)

def hasPhoto(self):
    return not self._empty

def fitInView(self, scale=True):
    rect = QtCore.QRectF(self._photo.pixmap().rect())
    if not rect.isNull():
        self.setSceneRect(rect)
        if self.hasPhoto():
            unity = self.transform().mapRect(QtCore.QRectF(0, 0, 1, 1))
            self.scale(1 / unity.width(), 1 / unity.height())
            viewrect = self.viewport().rect()
            scenerect = self.transform().mapRect(rect)
            factor = min(viewrect.width() / scenerect.width(),
                         viewrect.height() / scenerect.height())
            self.scale(factor, factor)
        self._zoom = 0
```

```
def setPhoto(self, pixmap=None):
    self._zoom = 0
    if pixmap and not pixmap.isNull():
        self._empty = False
        self.setDragMode(QtWidgets.QGraphicsView.ScrollHandDrag)
        self._photo.setPixmap(pixmap)

    else:
        self._empty = True
        self.setDragMode(QtWidgets.QGraphicsView.NoDrag)
        self._photo.setPixmap(QtGui.QPixmap())
    self.fitInView()

def wheelEvent(self, event):
    if self.hasPhoto():
        if event.angleDelta().y() > 0:
            factor = 1.25
            self._zoom += 1
        else:
            factor = 0.8
            self._zoom -= 1
        if self._zoom > 0:
            self.scale(factor, factor)
        elif self._zoom == 0:
            self.fitInView()
```

```
        else:
            self._zoom = 0

    def toggleDragMode(self):
        if self.dragMode() == QtWidgets.QGraphicsView.ScrollHandDrag:
            self.setDragMode(QtWidgets.QGraphicsView.NoDrag)
        elif not self._photo.pixmap().isNull():
            self.setDragMode(QtWidgets.QGraphicsView.ScrollHandDrag)

    def mousePressEvent(self, event):
        if self._photo.isUnderMouse():
            self.photoClicked.emit(self.mapToScene(event.pos()).toPoint())
        super(PhotoViewer, self).mousePressEvent(event)
```

A2. Script Ui_TFG

```
from PyQt5.QtCore import Qt
from PyQt5.QtGui import QImage, QPixmap, QPalette, QPainter, QScreen
from PyQt5.QtPrintSupport import QPrintDialog, QPrinter
from PyQt5.QtWidgets import QLabel, QSizePolicy, QScrollArea, QMessageBox, QMainWindow, QMenu, QAction, \
    qApp, QFileDialog, QApplication, QTableWidget, QTableWidgetItem
import pyautogui
from datetime import datetime
from PyQt5 import QtCore, QtGui, QtWidgets
from segmentada import Segmentada
```

```
import matplotlib.image as mpimg
import pyqtgraph as pg
from pyqtgraph import PlotWidget, plot


class Ui_TFG(QMainWindow, Segmentada):
    def __init__ (self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.segmentada = Segmentada() #heredamos de segmentada
        self.imageLabel = PhotoViewer(self)
        self.imageLabel_2 = PhotoViewer(self)


    def setupUi(self, TFG):
        TFG.setObjectName("TFG")
        TFG.resize(1366, 768)
        TFG.setMinimumSize(QtCore.QSize(1366, 760))
        TFG.setAutoFillBackground(False)
        TFG.setStyleSheet("background-image: url('dddef.jpg');\n"
"font: 75 14pt \"Segoe UI\";\n"
"\n"
"Qframe\n"
"{\n"
"background:#333\n"
"border-radius:60px\n"
"}")


        self.centralwidget = QtWidgets.QWidget(TFG)
```

```
        self.centralwidget.setObjectName("centralwidget")


        self.taula = QtWidgets.QTableWidget(self.centralwidget)
        self.taula.setGeometry(QtCore.QRect(30, 60, 311, 571))
        self.taula.setAutoFillBackground(True)
        self.taula.verticalHeader().setVisible(False)
        self.taula.setObjectName("taula")


        self.label_8 = QtWidgets.QLabel(self.centralwidget)
        self.label_8.setGeometry(QtCore.QRect(70, 10, 211, 41))
        font = QtGui.QFont()
        font.setFamily("Segoe UI")
        font.setPointSize(14)
        font.setBold(False)
        font.setItalic(False)
        font.setWeight(9)
        self.label_8.setFont(font)
        self.label_8.setStyleSheet("\n"
"QFrame\n"
"{\n"
"background:#008c00;\n"
"border-radius:15px;\n"
"}\n"
"\n"
"QLabel\n"
"{\n"
```

```
"color:white;\n"
"}")
        self.label_8.setAlignment(QtCore.Qt.AlignCenter)
        self.label_8.setObjectName("label_8")
        self.label_2 = QtWidgets.QLabel(self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(890, 10, 181, 41))
        font = QtGui.QFont()
        font.setFamily("Segoe UI")
        font.setPointSize(14)
        font.setBold(False)
        font.setItalic(False)
        font.setWeight(9)
        self.label_2.setFont(font)
        self.label_2.setStyleSheet("\n"
"QFrame\n"
"{\n"
"background:#333;\n"
"border-radius:10px;\n"
"}\n"
"\n"
"QLabel\n"
"{\n"
"color:white;\n"
"}")
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setObjectName("label_2")
```

```
        self.label_1 = QtWidgets.QLabel(self.centralwidget)
        self.label_1.setGeometry(QtCore.QRect(380, 10, 181, 41))
        font = QtGui.QFont()
        font.setFamily("Segoe UI")
        font.setPointSize(14)
        font.setBold(False)
        font.setItalic(False)
        font.setWeight(9)
        self.label_1.setFont(font)
        self.label_1.setStyleSheet("\n"
"QFrame\n"
"{\n"
"background:#333;\n"
"border-radius:10px;\n"
"}\n"
"\n"
"QLabel\n"
"{\n"
"color:white;\n"
"}")
        self.label_1.setAlignment(QtCore.Qt.AlignCenter)
        self.label_1.setObjectName("label_1")


        pg.setConfigOption('foreground', 'w')
        self.histogram = PlotWidget(self.centralwidget)
        self.histogram.setGeometry(QtCore.QRect(380, 450, 651, 211))
```

```
        self.histogram.setObjectName("histogram")
        self.histogram.setBackground((30, 30, 30))




        self.label_5 = QtWidgets.QLabel(self.centralwidget)
        self.label_5.setGeometry(QtCore.QRect(380, 400, 181, 41))
        font = QtGui.QFont()
        font.setFamily("Segoe UI")
        font.setPointSize(14)
        font.setBold(False)
        font.setItalic(False)
        font.setWeight(9)
        self.label_5.setFont(font)
        self.label_5.setStyleSheet("\n"
"QFrame\n"
"{\n"
"background:#333;\n"
"border-radius:10px;\n"
"}\n"
"\n"
"QLabel\n"
"{\n"
"color:white;\n"
"}")
        self.label_5.setAlignment(QtCore.Qt.AlignCenter)
```

```
        self.label_5.setObjectName("label_5")


        self.printer = QPrinter()
        self.scaleFactor = 0.0
        self.scrollArea = QtWidgets.QScrollArea(self.centralwidget)
        self.scrollArea.setGeometry(QtCore.QRect(380, 60, 451, 321))
        self.scrollArea.setWidgetResizable(True)
        self.scrollArea.setObjectName("scrollArea")
        self.scrollArea.setWidget(self.imageLabel)
        self.scrollArea.setVisible(True)


        self.scrollArea_2 = QtWidgets.QScrollArea(self.centralwidget)
        self.scrollArea_2.setGeometry(QtCore.QRect(890, 60, 451, 321))
        self.scrollArea_2.setWidgetResizable(True)
        self.scrollArea_2.setObjectName("scrollArea_2")
        self.scrollArea_2.setWidget(self.imageLabel_2)
        self.scrollArea_2.setVisible(True)


######################
#     BUTTONS


        self.clickbutt = QtWidgets.QPushButton(self.centralwidget)
        self.clickbutt.setGeometry(QtCore.QRect(650, 30, 101, 21))
        self.clickbutt.setStyleSheet("font: 10pt \"Lucida Sans Unicode\";\n"
"background: rgb(255, 0, 0);\n"
"border-radius:5px;\n"
```

```
"color:white;")
        icon9 = QtGui.QIcon()
        icon9.addPixmap(QtGui.QPixmap("iconos/mouse.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.clickbutt.setIcon(icon9)
        self.clickbutt.setObjectName("clickbutt")

###     BOTONES DE IMAGEN PROCESADA

        self.clickbutt_2 = QtWidgets.QPushButton(self.centralwidget)
        self.clickbutt_2.setGeometry(QtCore.QRect(1160, 30, 101, 21))
        self.clickbutt_2.setStyleSheet("font: 10pt \"Lucida Sans Unicode\";\n"
"background: rgb(255, 0, 0);\n"
"border-radius:5px;\n"
"color:white;")
        self.clickbutt_2.setIcon(icon9)
        self.clickbutt_2.setObjectName("clickbutt_2")

##### OTHER BUTTONS

        self.trashbutt = QtWidgets.QPushButton(self.centralwidget)
        self.trashbutt.setGeometry(QtCore.QRect(1290, 390, 51, 31))
        self.trashbutt.setStyleSheet("font: 10pt \"Lucida Sans Unicode\";\n"
"background:#333;\n"
"border-radius:5px;\n"
"color:white;")
        icon10 = QtGui.QIcon()
```

```
        icon10.addPixmap(QtGui.QPixmap("iconos/trash.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.trashbutt.setIcon(icon10)
        self.trashbutt.setIconSize(QtCore.QSize(20, 20))
        self.trashbutt.setObjectName("trashbutt")

        self.guardar_tabla = QtWidgets.QPushButton(self.centralwidget)
        self.guardar_tabla.setGeometry(QtCore.QRect(30, 640, 311, 21))
        font = QtGui.QFont()
        font.setFamily("Lucida Sans Unicode")
        font.setPointSize(10)
        font.setBold(False)
        font.setItalic(False)
        font.setWeight(50)
        self.guardar_tabla.setFont(font)
        self.guardar_tabla.setAutoFillBackground(False)
        self.guardar_tabla.setStyleSheet("font: 10pt \"Lucida Sans Unicode\";\n"
"background: #008c00;\n"
"border-radius:5px;\n"
"color:white;")
        icon11 = QtGui.QIcon()
        icon11.addPixmap(QtGui.QPixmap("iconos/guardar.png"), QtGui.QIcon.Normal,
QtGui.QIcon.Off)
        self.guardar_tabla.setIcon(icon11)
        self.guardar_tabla.setObjectName("guardar_tabla")

#       END BUTTONS
```

```
#      LABELfont


        self.co_1 = QtWidgets.QLabel(self.centralwidget)
        self.co_1.setGeometry(QtCore.QRect(760, 30, 71, 21))

        self.co_1.setAlignment(QtCore.Qt.AlignCenter)
        self.co_1.setStyleSheet("font: 75 10pt \"Segoe UI\";\n")
        self.co_1.setObjectName("co_1")

        self.co_2 = QtWidgets.QLabel(self.centralwidget)
        self.co_2.setGeometry(QtCore.QRect(1270, 30, 71, 21))

        self.co_2.setAlignment(QtCore.Qt.AlignCenter)
        self.co_2.setStyleSheet("font: 75 10pt \"Segoe UI\";\n")
        self.co_2.setObjectName("co_2")

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

        TFG.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(TFG)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 1366, 31))
        self.menubar.setObjectName("menubar")


        self.createAction()
        self.createMenus()
```

```
def open(self):
    options = QFileDialog.Options()
    # fileName = QFileDialog.getOpenFileName(self, "Open File", QDir.currentPath())
    fileName, _ = QFileDialog.getOpenFileName(self, 'QFileDialog.getOpenFileName()', '',
                                              'Images (*.png *.jpeg *.jpg *.bmp *.gif)', options=options)
    if fileName:
        image = QImage(fileName) #necesitamos una QImage para hacer el display en qpixmap

        self.img_op = mpimg.imread(fileName) # pero para editar la imagen es mejor convertirla a un
numpy array. Así pues,
        #será "self.img" la que pasaremos para las segmentaciones y filtros.

        if image.isNull():
            QMessageBox.information(self, "Image imageLabel", "Cannot load %s." % fileName)
            return

        self.imageLabel.setPhoto(QtGui.QPixmap(image))

        self.process = None


        #   ENABLE the actions that need an image
        self.printAct.setEnabled(True)

        self.actionOtsu.setEnabled(True)
```

```
        self.actionWatershed.setEnabled(True)
        self.actionQuikshift.setEnabled(True)
        self.actionFelzenszwalb.setEnabled(True)
        self.actioncontrast_log.setEnabled(True)
        self.actionContrast_gamma.setEnabled(True)
        self.actionAutom_tic.setEnabled(True)
        self.actionk_manual.setEnabled(True)
        self.actionHistograma.setEnabled(True)


        if len(self.img_op.shape) ==2:
            self.actionGrayscale.setEnabled(True)
        elif len(self.img_op.shape) == 3 or len(self.img_op.shape) == 4:
            self.actionRGB.setEnabled(True)
            self.actionGrayscale.setEnabled(True)


def print_(self):
    dialog = QPrintDialog(self.printer, self)
    if dialog.exec_():
        painter = QPainter(self.printer)
        rect = painter.viewport()
        size = self.imageLabel.pixmap().size()
        size.scale(rect.size(), Qt.KeepAspectRatio)
        painter.setViewport(rect.x(), rect.y(), size.width(), size.height())
        painter.setWindow(self.imageLabel.pixmap().rect())
```

```
        painter.drawPixmap(0, 0, self.imageLabel.pixmap())


    def shoot(self):
        date = datetime.now()
        filename = date.strftime('%Y-%m-%d_%H-%M-%S.jpg')
        p = QScreen.grabWindow(app.primaryScreen(), self.centralwidget.winId()).save(filename, 'png')
        mu = QMessageBox()


        if self.actionCatal.isChecked():
            mu.setText("Imatge: '" + filename + "' guardada correctament")
        if self.actionCastellano.isChecked():
            mu.setText("Imagen: '" + filename + "' guardada correctamente")
        if self.actionEnglish.isChecked():
            mu.setText("Image: '" + filename + "' has been saved successfully")


        mu.setWindowTitle("Warning")
        mu.setIcon(QMessageBox.Information)
        x = mu.exec_()



    def pixInfo(self):
        self.imageLabel.toggleDragMode()


    def photoClicked(self, pos):
        if self.imageLabel.dragMode()  == QtWidgets.QGraphicsView.NoDrag:
            self.co_1.setText('%d, %d' % (pos.x(), pos.y()))
```

```
def pixInfo2(self):
    self.imageLabel_2.toggleDragMode()

def photoClicked2(self, pos):
    if self.imageLabel_2.dragMode()  == QtWidgets.QGraphicsView.NoDrag:
        self.co_2.setText('%d, %d' % (pos.x(), pos.y()))



def guardar(self):
    date = datetime.now()
    filename = date.strftime('%Y-%m-%d_%H-%M-%S.jpg')
    p = QScreen.grabWindow(app.primaryScreen(), self.scrollArea_2.winId()).save(filename, 'png')
    mx = QMessageBox()

    if self.actionCatal.isChecked():
        mx.setText("Imatge: '" + filename + "' guardada correctament")
    if self.actionCastellano.isChecked():
        mx.setText("Imagen: '" + filename + "' guardada correctamente")
    if self.actionEnglish.isChecked():
        mx.setText("Image: '" + filename + "' has been saved successfully")

    mx.setWindowTitle("Warning")
    mx.setIcon(QMessageBox.Information)
    x = mx.exec_()
```

```
def about_aim(self):
    m = QMessageBox()
    if self.actionCatal.isChecked():
        m.setWindowTitle("OBJECTIU DE L'APLICACIÓ")
        m.setText("<p> L'aplicació té l'objectiu de proporcionar una "
                  "llibreria d'aplicacions de processament d'imatges biomèdiques amb entorn gràfic d'usuari,"
                  " gràcies al desenvolupament d'un conjunt d'eines de processament d'imatges biomèdiques. </p>"
                  "<p> Les eines inclouen algorismes de pre-processat, filtratge, detecció de contorns,"
                  " anàlisi de textura i segmentació d'objectes. L'eina està orientada a un ús en entorns"
                  " de recerca clínica i aplicacions de suport al diagnòstic.</p>")

    if self.actionCastellano.isChecked():
        m.setWindowTitle("OBJECTIVO DE LA APLICACIÓN")
        m.setText("<p> El programa tiene el objectivo de proporcionar una "
                  "libreria de aplicaciones de procesamiento de imágenes biomédicas con un entorno gráfico de usuario,"
                  " gracias al desarrollo de un conjunto de herramientas de procesamiento de imágenes biomédicas. </p>"
                  "<p> Las herramientas incluyen algoritmos de pre-procesado, filtrado, detección de contornos,"
                  " análisis de textura y segmentación de objetos. La herramienta está orientada a un uso en entornos"
                  " de investigación clínica y aplicaciones de apoyo al diagnóstico.</p>")

    if self.actionEnglish.isChecked():
        m.setWindowTitle("APPICATION'S AIM")
```

```
        m.setText("<p> The application aims to provide a "
                  "biomedical image processing library with graphical user environment (GUI), through"
                  " the development of a set of biomedical image processing tools. </p>"
                  "<p> The tools include pre-processing algorithms, filtering, contour detection,"
                  " texture analysis and object segmentation. The tool is use-oriented in "
                  " clinical research environments and diagnostic support applications.</p>")

    m.setIcon(QMessageBox.Information)
    y = m.exec_()

  def about_gmm(self):
    m = QMessageBox()
    if self.actionCatal.isChecked():
      m.setWindowTitle("GAUSSIAN MIXTURE MODEL")
      m.setText("<p> En aquesta aplicació s'han implementat dos tipus de segmentació GMM; "
                "per imatges grayscale i per imatges en RGB. </p>"
                "<p> Si la imatge original té format RGB podrà fer tant segmentació GMM-Grayscale"
                " com segmentació GMM-RGB. Però, en cas d'escollir una imatge original en escala de grisos"
                ", només es podrà dur a terme la segmentació GMM-grayscale.</p>")

    if self.actionCastellano.isChecked():
      m.setWindowTitle("GAUSSIAN MIXTURE MODEL")
      m.setText("<p> En esta aplicación se han implementado dos tipos de segmentación GMM; "
                "para imágenes grayscale y para imágenes en RGB. </p>"
                "<p> Si la imagen original tiene formato RGB podrá hacer tanto segmentación GMM-Grayscale"
```

```
            " como segmentación GMM-RGB. Pero, en caso de elegir una imagen original en escala
de grises"
            ", sólo se podrá llevar a cabo la segmentación GMM-grayscale.</p>")

    if self.actionEnglish.isChecked():
        m.setWindowTitle("GAUSSIAN MIXTURE MODEL")
        m.setText("<p> Two types of GMM segmentation have been implemented in this application; "
            "for grayscale images and RGB images. </p>"
            "<p> If the original image has RGB format both GMM-Grayscale segmentation"
            " and GMM-RGB segmentation can be chosen. Akin, in case of a grayscale original image"
            ", only GMM-grayscale segmentation will be allowed.</p>")

    m.setIcon(QMessageBox.Information)
    y = m.exec_()


def about_contrast(self):
    m = QMessageBox()
    if self.actionCatal.isChecked():
        m.setWindowTitle("SEGMENTACIÓ I CONTRAST")
        m.setText("<p> En el moment de fer la segmentació, l'aplicació tindrà en compte si s'ha fet"
            " un contrast previament a la imatge. En cas de ser així, la segmentació es farà sobre"
            " la imatge amb el contrast. </p>"
            "<p> En cas de voler eliminar el contrast per dur a terme la segmentació, "
            "cliqui sobre el botó de l'escombraria i, posteriorment, segmenti la imatge.</p>")
```

```
        if self.actionCastellano.isChecked():
            m.setWindowTitle("SEGMENTACIÓN I CONTRASTE")
            m.setText("<p> En el momento de hacer la segmentación, la aplicación tendrá en cuenta si se ha hecho"
                      " un contraste previamente a la imagen. En caso afirmativo, la segmentación se hará sobre"
                      " la imagen con contraste. </p>"
                      "<p> En caso de querer eliminar el contraste para llevar a cabo la segmentación, "
                      "clique sobre el botón de la basura y, posteriormente, segmente la imagen.</p>")

        if self.actionEnglish.isChecked():
            m.setWindowTitle("SEGMENTATION & CONTRAST")
            m.setText("<p> When segmenting the image, the app considers whether it has been applied a contrast"
                      " previously. If so, the segmentation will be done to"
                      " the filtered image. </p>"
                      "<p> In case you wish to remove the contrast to perform the segmentation "
                      "click on the rubbish button and then segment the image.</p>")

        m.setIcon(QMessageBox.Information)
        y = m.exec_()

    def about_table(self):
        m = QMessageBox()
        if self.actionCatal.isChecked():
            m.setWindowTitle("TAULA DE CARACTERÍSTIQUES")
            m.setText("<p> Les característiques extretes s'obtenen a partir de la funció 'regionprops' de Python."
```

```
" S'aplicarà aquesta funció sobre la última imatge processada, amb la segmentació corresponent. </p>"

"<p> En cas de voler veure les característiques amb una segmentació diferent,"

" segmenti altre cop la imatge i després actualitzi la taula.</p>")



if self.actionCastellano.isChecked():

    m.setWindowTitle("TABLA DE CARACTERÍSTICAS")

    m.setText("<p> Las características extraídas se obtienen a partir de la función 'regionprops' de Python."

" Se aplicará esta función sobre la última imagen procesada, con la segmentación correspondiente. </p>"

"<p> En caso de querer ver las características con una segmentación diferente,"

" segmente de nuevo la imagen y luego actualice la tabla.</p>")



if self.actionEnglish.isChecked():

    m.setWindowTitle("FEATURES TABLE")

    m.setText("<p> The extracted features are obtained from Python's 'regionprops' function."

" This function will be applied to the last image processed, with the corresponding segmentation. </p>"

"<p> In case you wish to see the characteristics with a different segmentation,"

" segment the image again and then update the table.</p>")


  m.setIcon(QMessageBox.Information)

  y = m.exec_()


def about_habilitar(self):
```

```
    m = QMessageBox()
    if self.actionCatal.isChecked():
      m.setWindowTitle("PAN / (X, Y)")
      m.setText("<p> Aquesta opció permet, quan està habilitada, moure la imatge a la finestra."
            " En cas contrari, mostra les coordenades del píxel al clicar sobre la imatge. </p>"
            "<p> Per passar d'un mode a l'altre simplement cliqui sobre el botó vermell.</p>")

    if self.actionCastellano.isChecked():
      m.setWindowTitle("PAN / (X, Y)")
      m.setText("<p> Esta opción permite, cuando está habilitada, mover la imagen en la ventana."
            " En caso contrario, muestra las coordenadas del píxel al clicar sobre la imagen. </p>"
            "<p> Para pasar de un modo a otro simplemente pulse sobre el botón rojo.</p>")

    if self.actionEnglish.isChecked():
      m.setWindowTitle("ENABLING")
      m.setText("<p> This option allows, when enabled, to move the image throughout the window"
            " When it is not enabled, it shows the pixel coordinates when clicking on the image."
            " To switch from one mode to another simply click on the red button. </p>")

    m.setIcon(QMessageBox.Information)
    y = m.exec_()

  def about_save(self):
    m = QMessageBox()
    if self.actionCatal.isChecked():
      m.setWindowTitle("GUARDAR IMATGES I CAPTURES DE PANTALLA")
```

```
        m.setText("<p> Les imatges i les captures de pantalla es guardaran amb el nom de la data i
l'hora"
                  " del moment en que s'ha fet. Trobarà les imatges a la mateixa carpeta on tingui"
                  " l'aplicació. </p>")



    if self.actionCastellano.isChecked():
        m.setWindowTitle("GUARDAR IMÁGENES Y CAPTURAS DE PANTALLA")
        m.setText("<p> Las imágenes y las capturas de pantalla se guardarán con el nombre de la fecha
y la hora"
                  " del momento en que se han hecho. Encontrará las imágenes en la misma carpeta donde
tenga"
                  " la aplicación. </p>")



    if self.actionEnglish.isChecked():
        m.setWindowTitle("SAVE IMAGES & SCREENSHOTS")
        m.setText("<p> The images and screenshots will be saved with the name of the date and time"
                  " of the moment when they have been taken. You will find the images in the same folder
where you have "
                  "the application. </p>")


    m.setIcon(QMessageBox.Information)
    y = m.exec_()


  def about_hist_gauss(self):
    m = QMessageBox()
    if self.actionCatal.isChecked():
```

```
        m.setWindowTitle("HISTOGRAMA AMB GAUSSIANES")

        m.setText("<p> Mostra l'histograma en grayscale de la imatge original i les gaussianes que s'han"

                  " trobat amb la segmentació GMM. A la llegenda s'indica la mitja i la covariància de cadascuna. </p>"

                  "<p> Opció només disponible per segmentació GMM-grayscale.</p>")



    if self.actionCastellano.isChecked():

        m.setWindowTitle("HISTOGRAMA CON GAUSIANAS")

        m.setText("<p> Muestra el histograma en grayscale de la imagen original y las gausianas que se han"

                  " encontrado con la segmentación GMM. En la leyenda se indica la media y la covarianza decada una. </p>"

                  "<p> Opción únicamente disponible para segmentación GMM-grayscale.</p>")



    if self.actionEnglish.isChecked():

        m.setWindowTitle("HISTOGRAM WITH GAUSSIAN CURVES")

        m.setText("<p> Displays the grayscale histogram of the original image and the Gaussian curves"

                  " found by the GMM segmentation. The legend indicates the mean and the covariance of each of them. </p>"

                  "<p> Only available for GMM-grayscale segmentation.</p>")


    m.setIcon(QMessageBox.Information)

    y = m.exec_()



  def conceptes(self):
```

```
    m = QMessageBox()


    if self.actionCatal.isChecked():
        m.setWindowTitle("CONCEPTES")
        m.setText("<p> Cliqui sobre els conceptes següents per saber més informació: </p>"
                "<p> <a href=\"https://www.python.org/\"  >Python</a>" "</p>"
                "<p> <a href=\"https://doc.qt.io/qtforpython/\"  >PyQt5</a>" "</p>"
                "<p>  <a  href=\"https://daedalus.umkc.edu/StatisticalMethods/histograms.html\"
>Histogram</a>" "</p>"
                "<p>  <a
href=\"https://micro.magnet.fsu.edu/primer/java/digitalimaging/processing/gamma/index.html\"
>Contrast</a>" "</p>"
                "<p>  <a  href=\"https://www.upf.edu/web/simbiosys/segmentation\"
>Segmentation</a>" "</p>"
                "<p> <a href=\"https://scikit-learn.org/stable/modules/mixture.html#gmm\"  >Gaussian
Mixture Model (GMM) </a>" "</p>")


    if self.actionCastellano.isChecked():
        m.setWindowTitle("CONCEPTOS")
        m.setText("<p> Clique sobre los conceptos siguientes para saber más información: </p>"
                "<p> <a href=\"https://www.python.org/\"  >Python</a>" "</p>"
                "<p> <a href=\"https://doc.qt.io/qtforpython/\"  >PyQt5</a>" "</p>"
                "<p>  <a  href=\"https://daedalus.umkc.edu/StatisticalMethods/histograms.html\"
>Histogram</a>" "</p>"
                "<p>  <a
href=\"https://micro.magnet.fsu.edu/primer/java/digitalimaging/processing/gamma/index.html\"
>Contrast</a>" "</p>"
                "<p>  <a  href=\"https://www.upf.edu/web/simbiosys/segmentation\"
>Segmentation</a>" "</p>"
                "<p> <a href=\"https://scikit-learn.org/stable/modules/mixture.html#gmm\"  >Gaussian
Mixture Model (GMM) </a>" "</p>")
```

```
    if self.actionEnglish.isChecked():
        m.setWindowTitle("CONCEPTS")
        m.setText("<p> Click on the links below for further information: </p>"
                "<p> <a href=\"https://www.python.org/\"  >Python</a>" "</p>"
                "<p> <a href=\"https://doc.qt.io/qtforpython/\"  >PyQt5</a>" "</p>"
                "<p>  <a  href=\"https://daedalus.umkc.edu/StatisticalMethods/histograms.html\"
>Histogram</a>" "</p>"
                "<p>                                                                    <a
href=\"https://micro.magnet.fsu.edu/primer/java/digitalimaging/processing/gamma/index.html\"
>Contrast</a>" "</p>"
                "<p>  <a  href=\"https://www.upf.edu/web/simbiosys/segmentation\"
>Segmentation</a>" "</p>"
                "<p> <a href=\"https://scikit-learn.org/stable/modules/mixture.html#gmm\"  >Gaussian
Mixture Model (GMM) </a>" "</p>")
    m.setIcon(QMessageBox.Information)
    y = m.exec_()
###################################
  def createAction(self):
    self.actionSeleccionar_imatge = QtWidgets.QAction(TFG)
    self.actionSeleccionar_imatge.triggered.connect(self.open)
    icon = QtGui.QIcon()
    icon.addPixmap(QtGui.QPixmap("iconos/image.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
    self.actionSeleccionar_imatge.setIcon(icon)
    self.actionSeleccionar_imatge.setObjectName("actionSeleccionar_imatge")
```

```
self.actionGuardar = QtWidgets.QAction(self, shortcut="Ctrl+S", triggered=self.guardar)
icon1 = QtGui.QIcon()
icon1.addPixmap(QtGui.QPixmap("iconos/guardar.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
self.actionGuardar.setIcon(icon1)
self.actionGuardar.setObjectName("actionGuardar")

self.actionScreenshot = QtWidgets.QAction(self, shortcut="Ctrl+G", triggered=self.shoot)
icon1 = QtGui.QIcon()
icon1.addPixmap(QtGui.QPixmap("iconos/scshot.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
self.actionScreenshot.setIcon(icon1)
self.actionScreenshot.setObjectName("actionScreenshot")

self.actionHistograma = QtWidgets.QAction("&Histograma", self, enabled=False,
triggered=self.histograma)
icon2 = QtGui.QIcon()
icon2.addPixmap(QtGui.QPixmap("iconos/plot.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
self.actionHistograma.setIcon(icon2)
self.actionHistograma.setObjectName("actionHistograma")

self.actionHistograma_amb_Gaussianes = QtWidgets.QAction(self, enabled=False,
triggered=self.hist_gauss)
icon3 = QtGui.QIcon()
icon3.addPixmap(QtGui.QPixmap("iconos/gmm.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
self.actionHistograma_amb_Gaussianes.setIcon(icon3)
self.actionHistograma_amb_Gaussianes.setObjectName("actionHistograma_amb_Gaussianes")
```

```
        self.actionContrast_gamma = QtWidgets.QAction("&Gamma", self, enabled=False, triggered=self.gammacorrection)

        icon4 = QtGui.QIcon()

        icon4.addPixmap(QtGui.QPixmap("iconos/gamma.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

        self.actionContrast_gamma.setIcon(icon4)

        self.actionContrast_gamma.setObjectName("actionContrast_gamma")


        self.actioncontrast_log = QtWidgets.QAction("&Logarítmic", self, enabled=False, triggered=self.logcorrection)

        icon5 = QtGui.QIcon()

        icon5.addPixmap(QtGui.QPixmap("iconos/log.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

        self.actioncontrast_log.setIcon(icon5)

        self.actioncontrast_log.setObjectName("actioncontrast_log")


        self.action_rea_Intensitat = QtWidgets.QAction("&Àrea/Intensitat", self, enabled=False, triggered=self.scatter)

        icon6 = QtGui.QIcon()

        icon6.addPixmap(QtGui.QPixmap("iconos/graph.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

        self.action_rea_Intensitat.setIcon(icon6)

        self.action_rea_Intensitat.setObjectName("action_rea_Intensitat")


        self.actionOtsu = QtWidgets.QAction("&Otsu", self, enabled=False, triggered=self.segm_otsu)

        self.actionOtsu.setObjectName("actionOtsu")


        self.actionWatershed = QtWidgets.QAction("&Watershed", self, enabled=False, triggered=self.segm_water)

        self.actionWatershed.setObjectName("actionWatershed")
```

```
        self.actionAutom_tic = QtWidgets.QAction("&Automàtic", self, enabled=False, triggered=self.segm_k)

        self.actionAutom_tic.setObjectName("actionAutom_tic")


        self.actionFelzenszwalb = QtWidgets.QAction("&Felzenszwalb", self, enabled=False, triggered=self.segm_felz)

        self.actionFelzenszwalb.setObjectName("actionFelzenszwalb")


        self.actionQuikshift = QtWidgets.QAction("&Quickshift", self, enabled=False, triggered=self.segm_quick)

        self.actionQuikshift.setObjectName("actionQuikshift")


        self.actionGrayscale = QtWidgets.QAction("&Imatge Grayscale", self, enabled=False, triggered=self.segmentacio_gauss_g)

        self.actionGrayscale.setObjectName("actionGrayscale")


        self.actionRGB = QtWidgets.QAction("&Imatge RGB", self, enabled=False, triggered=self.segmentacio_gauss_color)

        self.actionRGB.setObjectName("actionRGB")


        self.actionk_manual = QtWidgets.QAction("&Manual", self, enabled=False, triggered=self.manual)

        self.actionk_manual.setObjectName("actionk_manual")



        self.actionPropietats = QtWidgets.QAction(self, enabled=False, triggered=self.prop)

        self.actionPropietats.setObjectName("actionPropietats")
```

```
        self.clickbutt.clicked.connect(self.pixInfo)

        self.imageLabel.photoClicked.connect(self.photoClicked)



        self.clickbutt_2.clicked.connect(self.pixInfo2)

        self.imageLabel_2.photoClicked.connect(self.photoClicked2)


        self.guardar_tabla.clicked.connect(self.handleSave)

        self.guardar_tabla.setEnabled(False)


        self.printAct = QtWidgets.QAction("&Print...", self, shortcut="Ctrl+P", enabled=False,
triggered=self.print_)

        icon7 = QtGui.QIcon()

        icon7.addPixmap(QtGui.QPixmap("iconos/print.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

        self.printAct.setIcon(icon7)

        self.printAct.setObjectName("printAct")


        self.about_aim_Act  = QAction(self, triggered=self.about_aim)

        self.aboutQtAct = QAction("&Qt", self, triggered=qApp.aboutQt)


        self.about_gmm_act  = QAction(self, triggered=self.about_gmm)


        self.about_contrast_act  = QAction(self, triggered=self.about_contrast)


        self.about_table_act  = QAction(self, triggered=self.about_table)
```

```
self.about_habilitar_act  = QAction(self, triggered=self.about_habilitar)

self.about_save_act  = QAction(self, triggered=self.about_save)
self.about_hist_gauss_act =  QAction(self, triggered=self.about_hist_gauss)

self.actionCatal = QtWidgets.QAction(self, checkable=True, triggered=self.retranslateUi)
self.actionCatal.setChecked(True)
icon11 = QtGui.QIcon()
icon11.addPixmap(QtGui.QPixmap("iconos/cat.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
self.actionCatal.setIcon(icon11)
self.actionCatal.setObjectName("actionCatal")

self.actionCastellano = QtWidgets.QAction(self,checkable=True,
triggered=self.idioma_castellano)
icon12 = QtGui.QIcon()
icon12.addPixmap(QtGui.QPixmap("iconos/spain.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
self.actionCastellano.setIcon(icon12)
self.actionCastellano.setObjectName("actionCastellano")

self.actionEnglish = QtWidgets.QAction(self,checkable=True, triggered=self.idioma_english)
icon13 = QtGui.QIcon()
icon13.addPixmap(QtGui.QPixmap("iconos/eng.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
self.actionEnglish.setIcon(icon13)
self.actionEnglish.setObjectName("actionEnglish")
```

```
        self.trashbutt.clicked.connect(self.delete_process)
        if self.actionCatal.isChecked():
            self.trashbutt.setToolTip('Eliminar contrast')

        self.conceptes_act = QtWidgets.QAction(self, triggered=self.conceptes)

    def createMenus(self):
        self.menuFitxer = QtWidgets.QMenu(self.menubar)
        self.menuFitxer.setObjectName("menuFitxer")
        self.menuFitxer.addAction(self.actionSeleccionar_imatge)
        self.menuFitxer.addAction(self.actionGuardar)
        self.menuFitxer.addAction(self.printAct)
        self.menuFitxer.addAction(self.actionScreenshot)

        self.menuHistograma = QtWidgets.QMenu(self.menubar)
        self.menuHistograma.setObjectName("menuHistograma")
        self.menuHistograma.addAction(self.actionHistograma)
        self.menuHistograma.addAction(self.actionHistograma_amb_Gaussianes)

        self.menuProcessament = QtWidgets.QMenu(self.menubar)
        self.menuProcessament.setObjectName("menuProcessament")
        self.menuProcessament.addAction(self.actionContrast_gamma)
        self.menuProcessament.addAction(self.actioncontrast_log)
```

```
self.menuSegmentaci = QtWidgets.QMenu(self.menubar)
self.menuSegmentaci.setObjectName("menuSegmentaci")
self.menuSegmentaci.addAction(self.actionOtsu)
self.menuSegmentaci.addAction(self.actionWatershed)
self.menuSegmentaci.addAction(self.actionFelzenszwalb)
self.menuSegmentaci.addAction(self.actionQuikshift)

self.menuK_Means = QtWidgets.QMenu(self.menuSegmentaci)
self.menuSegmentaci.addAction(self.menuK_Means.menuAction())

self.menuGMM = QtWidgets.QMenu(self.menuSegmentaci)
self.menuSegmentaci.addAction(self.menuGMM.menuAction())

self.menuK_Means.setObjectName("menuK_Means")
self.menuK_Means.addAction(self.actionAutom_tic)
self.menuK_Means.addAction(self.actionk_manual)



self.menuGMM.setObjectName("menuGMM")
self.menuGMM.addAction(self.actionGrayscale)
self.menuGMM.addAction(self.actionRGB)

self.menuPropietats = QtWidgets.QMenu(self.menubar)
self.menuPropietats.setObjectName("menuPropietats")
```

```
self.menuPropietats.addAction(self.actionPropietats)
if self.actionCatal.isChecked():
   self.menuPropietats.setToolTip('Sobre la última segmentació realitzada')



self.menuGr_fic = QtWidgets.QMenu(self.menubar)
self.menuGr_fic.setObjectName("menuGr_fic")
self.menuGr_fic.addAction(self.action_rea_Intensitat)




self.menuIdioma = QtWidgets.QMenu(self.menubar)
self.menuIdioma.setObjectName("menuIdioma")
self.menuIdioma.addAction(self.actionCatal)
self.menuIdioma.addAction(self.actionCastellano)
self.menuIdioma.addAction(self.actionEnglish)

self.menuAjuda = QtWidgets.QMenu(self.menubar)
self.menuAjuda.setObjectName("menuAjuda")
self.menuAjuda.addAction(self.aboutQtAct)
self.menuAjuda.addAction(self.conceptes_act)

self.MenuAbout = QtWidgets.QMenu(self.menuAjuda)
self.menuAjuda.addAction(self.MenuAbout.menuAction())
self.MenuAbout.addAction(self.about_aim_Act)
self.MenuAbout.addAction(self.about_gmm_act)
```

```
self.MenuAbout.addAction(self.about_contrast_act)
self.MenuAbout.addAction(self.about_table_act)
self.MenuAbout.addAction(self.about_habilitar_act)
self.MenuAbout.addAction(self.about_save_act)
self.MenuAbout.addAction(self.about_hist_gauss_act)

self.menubar.addAction(self.menuFitxer.menuAction())

self.menubar.addAction(self.menuHistograma.menuAction())
self.menubar.addAction(self.menuProcessament.menuAction())
self.menubar.addAction(self.menuSegmentaci.menuAction())
self.menubar.addAction(self.menuPropietats.menuAction())
self.menubar.addAction(self.menuGr_fic.menuAction())
self.menubar.addAction(self.menuAjuda.menuAction())
self.menubar.addAction(self.menuIdioma.menuAction())



TFG.setMenuBar(self.menubar)
self.retranslateUi(TFG)
QtCore.QMetaObject.connectSlotsByName(TFG)
_translate = QtCore.QCoreApplication.translate
TFG.setWindowTitle(_translate("TFG", "TFG"))
self.actionCatal.setChecked(True)
self.actionCastellano.setChecked(False)
self.actionEnglish.setChecked(False)
```

```
def retranslateUi(self, TFG):
    self.actionCatal.setChecked(True)
    self.actionCastellano.setChecked(False)
    self.actionEnglish.setChecked(False)

    self.trashbutt.setToolTip('Eliminar contrast')
    self.menuPropietats.setToolTip('Sobre la última segmentació realitzada')

    _translate = QtCore.QCoreApplication.translate
    self.about_hist_gauss_act.setText(_translate("TFG", "Histograma amb Gaussianes"))
    self.conceptes_act.setText(_translate("TFG", "Conceptes"))
    self.about_aim_Act.setText(_translate("TFG", "Objectiu"))
    self.about_gmm_act.setText(_translate("TFG", "GMM"))
    self.about_contrast_act.setText(_translate("TFG", "Contrast"))
    self.about_table_act.setText(_translate("TFG", "Taula de característiques"))
    self.about_habilitar_act.setText(_translate("TFG", "Pan / zoom"))
    self.about_save_act.setText(_translate("TFG", "Guardar imatges"))
    self.MenuAbout.setTitle(_translate("TFG", "L'aplicació"))
    self.label_8.setText(_translate("TFG", "Taula de característiques"))
    self.label_2.setText(_translate("TFG", "Imatge Processada"))
    self.label_1.setText(_translate("TFG", "Imatge Original"))
    self.label_5.setText(_translate("TFG", "Histograma"))

    self.clickbutt.setText(_translate("TFG", "Pan / (x , y)"))
    self.clickbutt_2.setText(_translate("TFG", "Pan / (x , y)"))
```

```
self.guardar_tabla.setText(_translate("TFG", "Guardar dades"))
self.co_1.setText('(x , y)')
self.co_2.setText('(x , y)')
self.menuFitxer.setTitle(_translate("TFG", "Fitxer"))

self.menuHistograma.setTitle(_translate("TFG", "Histograma"))
self.menuProcessament.setTitle(_translate("TFG", "Contrast"))
self.menuSegmentaci.setTitle(_translate("TFG", "Segmentació"))
self.menuK_Means.setTitle(_translate("TFG", "K-Means"))
self.actionk_manual.setText(_translate("TFG", "Manual"))
self.menuGMM.setTitle(_translate("TFG", "GMM"))
self.menuPropietats.setTitle(_translate("TFG", "Propietats"))

self.menuGr_fic.setTitle(_translate("TFG", "Gràfic"))
self.menuAjuda.setTitle(_translate("TFG", "Ajuda"))
self.clickbutt.setShortcut(_translate("TFG", "Ctrl+M"))
self.clickbutt_2.setShortcut(_translate("TFG", "Ctrl+O"))
self.menuIdioma.setTitle(_translate("TFG", "Idioma"))
self.actionSeleccionar_imatge.setText(_translate("TFG", "Seleccionar imatge"))
self.actionSeleccionar_imatge.setShortcut(_translate("TFG", "Ctrl+F"))
self.actionGuardar.setText(_translate("TFG", "Guardar imatge processada"))
self.actionGuardar.setShortcut(_translate("TFG", "Ctrl+S"))
self.actionScreenshot.setText(_translate("TFG", "Captura de pantalla"))
self.actionHistograma.setText(_translate("TFG", "Histograma"))
```

```
        self.actionHistograma_amb_Gaussianes.setText(_translate("TFG", "Histograma amb
Gaussianes"))
        self.actionContrast_gamma.setText(_translate("TFG", "Gamma"))
        self.actioncontrast_log.setText(_translate("TFG", "Logarítmic"))
        self.action_rea_Intensitat.setText(_translate("TFG", "Àrea / Intensitat"))
        self.actionOtsu.setText(_translate("TFG", "Otsu"))
        self.actionWatershed.setText(_translate("TFG", "Watershed"))
        self.actionAutom_tic.setText(_translate("TFG", "Automàtic"))
        self.actionFelzenszwalb.setText(_translate("TFG", "Felzenszwalb"))
        self.actionQuikshift.setText(_translate("TFG", "Quikshift"))
        self.actionGrayscale.setText(_translate("TFG", "Grayscale"))
        self.actionRGB.setText(_translate("TFG", "RGB"))
        self.actionPropietats.setText(_translate("TFG", "Taula"))
        self.printAct.setText(_translate("TFG", "Imprimir"))

        self.actionCatal.setText(_translate("TFG", "Català"))
        self.actionCastellano.setText(_translate("TFG", "Español"))
        self.actionEnglish.setText(_translate("TFG", "English"))

        if self.histogram_is_done is not None:
            self.hist_lang()

        if self.table_is_done is not None:
            self.prop_lang()

        if self.graph_is_done is not None:
```

```
        self.scatter()

    if self.histogram_gauss_is_done is not None:
        self.hist_lang()



def idioma_castellano(self, TFG):

    _translate = QtCore.QCoreApplication.translate
    self.actionCatal.setChecked(False)
    self.actionCastellano.setChecked(True)
    self.actionEnglish.setChecked(False)
    self.menuPropietats.setToolTip('Sobre la última segmentación realitzada')
    self.conceptes_act.setText(_translate("TFG", "Conceptos"))
    self.about_aim_Act.setText(_translate("TFG", "Objetivo"))
    self.about_gmm_act.setText(_translate("TFG", "GMM"))
    self.about_contrast_act.setText(_translate("TFG", "Contraste"))
    self.about_table_act.setText(_translate("TFG", "Tabla de características"))
    self.about_habilitar_act.setText(_translate("TFG", "Pan / zoom"))
    self.about_save_act.setText(_translate("TFG", "Guardar imágenes"))
    self.MenuAbout.setTitle(_translate("TFG", "La aplicación"))
    self.clickbutt.setText(_translate("TFG", "Pan / (x , y)"))
    self.clickbutt_2.setText(_translate("TFG", "Pan / (x , y)"))
    self.label_8.setText(_translate("TFG", "Tabla de características"))
    self.label_2.setText(_translate("TFG", "Imagen Procesada"))
```

```
self.label_1.setText(_translate("TFG", "Imagen Original"))
self.label_5.setText(_translate("TFG", "Histograma"))


self.guardar_tabla.setText(_translate("TFG", "Guardar datos"))

self.menuFitxer.setTitle(_translate("TFG", "Fichero"))

self.menuHistograma.setTitle(_translate("TFG", "Histograma"))
self.menuProcessament.setTitle(_translate("TFG", "Contraste"))
self.menuSegmentaci.setTitle(_translate("TFG", "Segmentación"))
self.menuK_Means.setTitle(_translate("TFG", "K-Means"))
self.actionk_manual.setText(_translate("TFG", "Manual"))
self.menuGMM.setTitle(_translate("TFG", "GMM"))
self.menuPropietats.setTitle(_translate("TFG", "Propiedades"))

self.menuGr_fic.setTitle(_translate("TFG", "Gráfico"))
self.menuAjuda.setTitle(_translate("TFG", "Ayuda"))
self.menuIdioma.setTitle(_translate("TFG", "Idioma"))
self.actionSeleccionar_imatge.setText(_translate("TFG", "Seleccionar imagen"))

self.actionGuardar.setText(_translate("TFG", "Guardar imagen procesada"))

self.actionScreenshot.setText(_translate("TFG", "Captura de pantalla"))
self.actionHistograma.setText(_translate("TFG", "Histograma"))
self.about_hist_gauss_act.setText(_translate("TFG", "Histograma con Gausianas"))
```

```
self.actionHistograma_amb_Gaussianes.setText(_translate("TFG", "Histograma con Gausianas"))
self.actionContrast_gamma.setText(_translate("TFG", "Gamma"))
self.actioncontrast_log.setText(_translate("TFG", "Logarítmico"))
self.action_rea_Intensitat.setText(_translate("TFG", "Área / Intensidad"))
self.actionOtsu.setText(_translate("TFG", "Otsu"))
self.actionWatershed.setText(_translate("TFG", "Watershed"))
self.actionAutom_tic.setText(_translate("TFG", "Automático"))
self.actionFelzenszwalb.setText(_translate("TFG", "Felzenszwalb"))
self.actionQuikshift.setText(_translate("TFG", "Quikshift"))
self.actionGrayscale.setText(_translate("TFG", "Grayscale"))
self.actionRGB.setText(_translate("TFG", "RGB"))
self.actionPropietats.setText(_translate("TFG", "Tabla"))
self.printAct.setText(_translate("TFG", "Imprimir"))

self.trashbutt.setToolTip('Eliminar contraste')

if self.histogram_is_done is not None:
    self.hist_lang()

if self.table_is_done is not None:
    self.prop_lang()

if self.graph_is_done is not None:
    self.scatter()

if self.histogram_gauss_is_done is not None:
```

```
        self.hist_lang()


    def idioma_english(self, TFG):


        _translate = QtCore.QCoreApplication.translate
        self.actionCatal.setChecked(False)
        self.actionCastellano.setChecked(False)
        self.actionEnglish.setChecked(True)
        self.menuPropietats.setToolTip('Of the last segmentation')
        self.conceptes_act.setText(_translate("TFG", "Concepts"))
        self.about_aim_Act.setText(_translate("TFG", "Aim"))
        self.about_gmm_act.setText(_translate("TFG", "GMM"))
        self.about_contrast_act.setText(_translate("TFG", "Contrast"))
        self.about_table_act.setText(_translate("TFG", "Features table"))
        self.about_habilitar_act.setText(_translate("TFG", "Pan / zoom"))
        self.about_save_act.setText(_translate("TFG", "Save images"))
        self.MenuAbout.setTitle(_translate("TFG", "the application"))
        self.label_8.setText(_translate("TFG", "Features table"))
        self.label_2.setText(_translate("TFG", "Processed image"))
        self.label_1.setText(_translate("TFG", "Original Image"))
        self.label_5.setText(_translate("TFG", "Histogram"))
        self.clickbutt.setText(_translate("TFG", "Pan / (x , y)"))
        self.clickbutt_2.setText(_translate("TFG", "Pan / (x , y)"))



        self.guardar_tabla.setText(_translate("TFG", "Save data"))
```

```
self.menuFitxer.setTitle(_translate("TFG", "File"))

self.menuHistograma.setTitle(_translate("TFG", "Histogram"))
self.menuProcessament.setTitle(_translate("TFG", "Contrast"))
self.menuSegmentaci.setTitle(_translate("TFG", "Segmentation"))
self.menuK_Means.setTitle(_translate("TFG", "K-Means"))
self.actionk_manual.setText(_translate("TFG", "Manual"))
self.menuGMM.setTitle(_translate("TFG", "GMM"))
self.menuPropietats.setTitle(_translate("TFG", "Features"))

self.menuGr_fic.setTitle(_translate("TFG", "Graph"))
self.menuAjuda.setTitle(_translate("TFG", "Help"))
self.menuIdioma.setTitle(_translate("TFG", "Language"))
self.actionSeleccionar_imatge.setText(_translate("TFG", "Select image"))

self.actionGuardar.setText(_translate("TFG", "Save processed image"))

self.actionScreenshot.setText(_translate("TFG", "Screenshot"))
self.actionHistograma.setText(_translate("TFG", "Histogram"))
self.about_hist_gauss_act.setText(_translate("TFG", "Histogram with Gaussians"))
self.actionHistograma_amb_Gaussianes.setText(_translate("TFG", "Histogram with Gaussians"))
self.actionContrast_gamma.setText(_translate("TFG", "Gamma"))
self.actioncontrast_log.setText(_translate("TFG", "Logarithmic"))
self.action_rea_Intensitat.setText(_translate("TFG", "Area / Intensity"))
self.actionOtsu.setText(_translate("TFG", "Otsu"))
```

```
        self.actionWatershed.setText(_translate("TFG", "Watershed"))
        self.actionAutom_tic.setText(_translate("TFG", "Automatic"))
        self.actionFelzenszwalb.setText(_translate("TFG", "Felzenszwalb"))
        self.actionQuikshift.setText(_translate("TFG", "Quikshift"))
        self.actionGrayscale.setText(_translate("TFG", "Grayscale"))
        self.actionRGB.setText(_translate("TFG", "RGB"))
        self.actionPropietats.setText(_translate("TFG", "Table"))
        self.printAct.setText(_translate("TFG", "Print"))



        self.trashbutt.setToolTip('Remove contrast filter')

        if self.histogram_is_done is not None:
            self.hist_lang()

        if self.table_is_done is not None:
            self.prop_lang()

        if self.graph_is_done is not None:
            self.scatter()

        if self.histogram_gauss_is_done is not None:
            self.hist_lang()

import para_fotos
```

```
if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    TFG = QtWidgets.QMainWindow()
    ui = Ui_TFG()
    ui.setupUi(TFG)
    TFG.show()
    sys.exit(app.exec_())
print(ui.__dict__.keys())
```

A3. Script Segmentada

```
from skimage import io, exposure
from skimage.filters import threshold_otsu, sobel
from skimage.segmentation import clear_border, watershed, felzenszwalb, slic, quickshift
from sklearn.cluster import KMeans, AgglomerativeClustering
from skimage.measure import label
from skimage.morphology import closing, square
from skimage.transform import resize
from skimage.color import label2rgb, rgb2gray
import numpy as np
from sklearn import mixture
from skimage.morphology import disk
from skimage.filters import median, gaussian
import skimage as sk
```

```
import itertools

from scipy import linalg

import scipy.stats as stats

import math

import pylab as pl

import scipy as scipy

from scipy.spatial import distance

from PyQt5.QtGui import QImage, QPixmap

from PyQt5 import QtCore, QtGui, QtWidgets

import matplotlib.pyplot as plt

#import PySide2.QtGui as qtg

import numpy as np

from PyQt5.QtGui import QImage, qRgb

from PyQt5.QtWidgets import QMessageBox, QInputDialog, QTableWidgetItem

import pyqtgraph as pg

from skimage.measure import regionprops

import pandas as pd

from PyQt5.QtWidgets import QFileDialog

import matplotlib.pyplot as plt
```

```
class Segmentada:

  def __init__(self, process=None, label = None, imaori = None, histogram_is_done=None, table_is_done=None, graph_is_done=None, histogram_gauss_is_done=None):

    self.process = process
```

```
        self.label = label
        self.imaori = imaori
        self.histogram_is_done = histogram_is_done
        self.table_is_done = table_is_done
        self.graph_is_done = graph_is_done
        self.histogram_gauss_is_done = histogram_gauss_is_done


    def pre_segm(self,img_op):
        self.actionPropietats.setEnabled(True)
        if self.process is not None: #si ya se ha hecho una corrección que utilice esa imagen para empezar, no la original
            self.imaori = self.process
        else:
            self.imaori=rgb2gray(self.img_op)


    def toQImage(self, im):
        gray_color_table = [qRgb(0, i, 0) for i in range(256)]


        if im is None:
            return QImage()


        im_255 = im*255 #float64 define el rango de [0-255] de grayscale como [0-1] con decimales, hay que multiplicar por 255.
        img_8 = im_255.astype(np.uint8) #entonces la convertimos a int8


        if img_8.dtype == np.uint8:
```

```
        img_8 = np.require(img_8, np.uint8, 'C')


        if len(img_8.shape) == 2: #dependiendo del número de canales que tenga, significará que la imagen es en grayscale, RGB o ARGB

            self.qim = QImage(img_8.data, img_8.shape[1], img_8.shape[0], img_8.strides[0], QImage.Format_Indexed8)#transformamos a QImage

            self.qim.setColorTable(gray_color_table)



        elif len(img_8.shape) == 3: #si el shape de la I es 3, es porque tenemos (x_pix, y_pix, canales)

            if img_8.shape[2] == 3: #ahora podemos tener 3 canales (RGB)

              self.qim = QImage(img_8.data, img_8.shape[1], img_8.shape[0], img_8.strides[0], QImage.Format_RGB888);



            elif img_8.shape[2] == 4: #o tener 4 canales (ARGB)

              self.qim = QImage(img_8.data, img_8.shape[1], img_8.shape[0], img_8.strides[0], QImage.Format_ARGB32);



      self.imageLabel_2.setPhoto(QtGui.QPixmap(self.qim)) #qim es la imagen ya en qImage



  def delete_process(self):

      self.process = None

      msg = QMessageBox()

      if self.actionCatal.isChecked():

        msg.setText("Contrast eliminat! " "Torna a segmentar")
```

```
        if self.actionCastellano.isChecked():
            msg.setText("Contraste eliminado! " "Vuelve a segmentar")
        if self.actionEnglish.isChecked():
            msg.setText("Contrast removed! " "Repeat the segmentation")
        msg.setWindowTitle("Warning")
        msg.setIcon(QMessageBox.Warning)
        x = msg.exec_()

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


    def gammacorrection(self, img_op):
        self.actionHistograma_amb_Gaussianes.setEnabled(False)
        self.imaori=rgb2gray(self.img_op)#para la corrección gamma primero pasamos la imagen de rgb a escala de grises
        self.process = exposure.adjust_gamma(self.imaori, 2)#ajustamos la exposición según el ajuste gamma
        self.toQImage(self.process) #convertimos la imagen de formato float64 a QImage para poder visualizarla, mediante una función


    def logcorrection(self, img_op):
        self.actionHistograma_amb_Gaussianes.setEnabled(False)
        self.imaori=rgb2gray(self.img_op)
        self.process = exposure.adjust_log(self.imaori, 1)#hace lo mismo pero con ajuste logarítmico
        self.toQImage(self.process)
```

##33

```
def segm_otsu(self, img_op):
    self.actionHistograma_amb_Gaussianes.setEnabled(False)
    self.pre_segm(self.img_op)
    self.thresh = threshold_otsu(self.imaori)
    bw = closing(self.imaori < self.thresh, square(3))
    cleared = clear_border(bw)
    self.label = label(cleared)
    self.segmen = label2rgb(self.label, image=self.imaori)
    self.toQImage(self.segmen)


def segm_water(self,img_op):
    self.actionHistograma_amb_Gaussianes.setEnabled(False)
    self.pre_segm(self.img_op)
    gradient = sobel(self.imaori)
    self.label = watershed(gradient, markers=600)
    self.segmen=label2rgb(self.label, self.imaori)
    self.toQImage(self.segmen)


def segm_k(self,img_op): #detecta automàticament el número de clusters que vol escollir
    self.actionHistograma_amb_Gaussianes.setEnabled(False)
    self.pre_segm(self.img_op)
```

```
        self.label = slic(self.imaori, compactness=0.2, n_segments=600)
        self.segmen = label2rgb(self.label, self.imaori, kind='avg')
        self.toQImage(self.segmen)


    def segm_felz(self,img_op):
        self.actionHistograma_amb_Gaussianes.setEnabled(False)
        self.pre_segm(self.img_op)
        self.label = felzenszwalb(self.imaori, scale=100, sigma=0.5, min_size=50)
        self.segmen = label2rgb(self.label, self.imaori, kind='avg')
        self.toQImage(self.segmen)


    def segm_quick(self,img_op):
        self.actionHistograma_amb_Gaussianes.setEnabled(False)
        self.pre_segm(self.img_op)
        self.label = quickshift(self.imaori, kernel_size=5, ratio=1.0, max_dist=10, return_tree=False,
sigma=0, convert2lab=False, random_seed=42)
        self.segmen = label2rgb(self.label, self.imaori, kind='avg')
        self.toQImage(self.segmen)


    def segm_k_manual(self,img_op, n_clusters):
        self.actionHistograma_amb_Gaussianes.setEnabled(False)
        self.pre_segm(self.img_op)
        v_I = self.imaori.reshape(self.imaori.shape[0]*self.imaori.shape[1],1)
        km = KMeans(n_clusters=self.n_clusters).fit(v_I)
```

```
        pred = km.predict(v_I)

        v_agg= np.logical_not(pred)

        imafin = np.array(v_agg).reshape(self.imaori.shape[0],self.imaori.shape[1]) # Shape back result as
image

        self.label = label(imafin)

        self.segmen = label2rgb(self.label, self.imaori, kind='avg')

        self.toQImage(self.segmen)


    def manual(self, img_op):

        if self.actionCatal.isChecked() or self.actionCastellano.isChecked():

            self.n_clusters, result = QInputDialog.getInt(self,'Input','Número de clusters') #Diàlog on el
usuari especifica el número de clusters


        if self.actionEnglish.isChecked():

            self.n_clusters, result = QInputDialog.getInt(self,'Input','Number of clusters')




        if result == True:

            if 1<self.n_clusters<51 or type(self.n_clusters) != int:

                self.segm_k_manual(self.img_op, self.n_clusters)

            else:

                ms = QMessageBox()

                if self.actionCatal.isChecked():

                    ms.setText("El número de clusters ha de ser un enter al rang [2,50]")

                if self.actionCastellano.isChecked():

                    ms.setText("El número de clusters debe ser un entero en el rango [2,50]")

                if self.actionEnglish.isChecked():
```

```
            ms.setText("The number of clusters must be an integrer in range [2,50]")


        ms.setWindowTitle("Error")
        ms.setIcon(QMessageBox.Critical)
        y = ms.exec_()



  def segmentacio_gauss_g(self,img_op):
    self.actionHistograma_amb_Gaussianes.setEnabled(True)
    self.pre_segm(self.img_op)
    imaorif = median(self.imaori, disk(1))
    self.imaori = median(imaorif, disk(2))
    v_I_train = np.reshape(self.imaori,(self.imaori.shape[0]*self.imaori.shape[1],1)) #convertir a
vector
    #self.v_I_train = v_I_train
    #calcula el BIC per trobar el número de gaussianes òptim
    NMAX = 10
    bic = []
    for kG in np.arange(1,NMAX+1):
      gmm = mixture.GaussianMixture(n_components=kG).fit(v_I_train)
      bic.append(gmm.bic(v_I_train)) #cada cop va afegint el bic amb kG+1, així ho tens tot en un
vector i pots calcualr el mínim


    idx_winner = np.argmin(bic)
    gmmw = mixture.GaussianMixture(n_components=idx_winner).fit(v_I_train)
    means0= gmmw.means_.reshape(1,-1)
    means = np.array(means0).ravel()
```

```
M = np.zeros([len(means),2])
for x in range (0,len(means)):
   M[x][0] = means[x]
   M[x][1] = gmmw.weights_[x]

sortedm = M[M[:,0].argsort()]
means_def = []
x = 0
while x <=len(means)-2:
   if sortedm[x,0]+10<sortedm[x+1,0]:
      means_def.append(sortedm[x,0]) #en cas de que no estiguin aprop, l'afegim directament
      x = x+1
      if x==len(means)-1:
         means_def.append(sortedm[x,0]) #si el penúltim average i el últim no estan a prop, com que per l'últim ja no podem
         #seguir comparant amb cap altre següent, simplement l'afegim. De fet no fa falta comparar ja que ja ha sigut comparat
   else: #si estan a prop hem de mirar quin té un pes més alt, i aquest serà el que es quedi.
      if sortedm[x,1]>sortedm[x+1,1]:
         means_def.append(sortedm[x,0])
         x = x+2 #saltem dues posicions perquè la següent ja sabem que no és rellevant, està a prop d'aquesta i té un pes inferior.
         if x == len(means)-1:
            means_def.append(sortedm[x,0])
      else:
         means_def.append(sortedm[x+1,0])
```

```
            x = x+1


        means_def = np.unique(means_def)

        means_def = np.reshape(means_def,[len(means_def),1])

        self.gmw2 = mixture.GaussianMixture(n_components=len(means_def),
means_init=means_def).fit(v_I_train)

        v_agg = self.gmw2.fit_predict(v_I_train)

        imafin = np.array(v_agg).reshape(self.imaori.shape[0],self.imaori.shape[1])# Shape back result as
image

        self.label = label(imafin)

        self.segmen = label2rgb(imafin, self.imaori, kind='avg')

        self.toQImage(self.segmen)



    def segmentacio_gauss_color(self,img_op):

        self.actionHistograma_amb_Gaussianes.setEnabled(False)

        self.pre_segm(self.img_op)

        R = self.imaori[:,:,0]

        G = self.imaori[:,:,1]

        B = self.imaori[:,:,2]

        v_R = np.reshape(R,(R.shape[0]*R.shape[1],1))

        v_G = np.reshape(G,(G.shape[0]*G.shape[1],1))

        v_B = np.reshape(B,(B.shape[0]*B.shape[1],1))

        MT = np.zeros([len(v_R),3])

        for x in range (0,len(v_R)):

            MT[x,0] = v_R[x]

            MT[x,1] = v_G[x]
```

```
    MT[x,2] = v_B[x]
v_I_train = MT
lowest_bic = np.infty
bic = []
n_components_range = range(1, 11)
cv_types = ['spherical', 'tied', 'diag', 'full']
for cv_type in cv_types:
    for n_components in n_components_range:
        # Fit a Gaussian mixture with EM
        gmm = mixture.GaussianMixture(n_components=n_components,
                                      covariance_type=cv_type)
        gmm.fit(v_I_train)
        bic.append(gmm.bic(v_I_train))
        if bic[-1] < lowest_bic:
            lowest_bic = bic[-1]
            best_gmm = gmm

bic = np.array(bic)
gmmw = best_gmm # GUANYADOR
bars = []
M = np.zeros([np.size(gmmw.means_[:,0]),4])
g = [[255,255,255]]
p = [[0,0,0]]
maxim = distance.cdist(g, p, 'euclidean')
eucl = distance.cdist(gmmw.means_, gmmw.means_, 'euclidean')
eucl_porc = eucl*100/maxim
```

```
means = gmmw.means_
todel = [] #gauss to delete
for i in range (0,np.size(eucl_porc[0])):
  for j in range (0,np.size(eucl_porc[1])):
    if eucl_porc[i,j] <= 10 and eucl_porc[i,j]!=0:
      if gmmw.weights_[i]>gmmw.weights_[j]:
        todel.append(j)
      else:
        todel.append(i)

todel = np.unique(todel)
means_def = []
for y in range (0,10):
  if y not in todel:
    means_def.append(means[y,:])
print(means_def)
gmw2 = mixture.GaussianMixture(n_components=len(means_def), means_init=means_def,
covariance_type=gmmw.covariance_type).fit(v_I_train)
v_agg2 = gmw2.predict(MT) #recorda que v_I_train es la matriu MT
imafin = np.array(v_agg2).reshape(imaori.shape[0],imaori.shape[1])
self.label = label(imafin)
self.segmen = label2rgb(imafin, self.imaori, kind='avg')
self.toQImage(self.segmen)

##################################################3333
```

```
    def histograma(self, img_op):

        self.pre_segm(self.img_op)

        v_I_train = (np.reshape(self.imaori,(self.imaori.shape[0]*self.imaori.shape[1],1)))*255 #convertir
a vector

        y,x = np.histogram(v_I_train, bins=256)

        pen = pg.mkBrush('c')

        self.histogram_is_done = 1


        y = y*100/np.sum(y)


        self.hist_lang()


        self.histogram.plot(x, y, stepMode=True, fillLevel=0, brush=pen)


    def hist_lang(self):
        if self.actionCatal.isChecked():
            self.histogram.setLabel('left', 'Percentatge de píxels (%)', color='white', size=50)
            self.histogram.setLabel('bottom', 'Nivell de píxel [0-255]')


        elif self.actionCastellano.isChecked():
            self.histogram.setLabel('left', 'Porcentaje de píxeles (%)', color='white', size=50)
            self.histogram.setLabel('bottom', 'Nivel de píxel [0-255]')


        elif self.actionEnglish.isChecked():
            self.histogram.setLabel('left', 'Pixel percentage (%)', color='white', size=50)
            self.histogram.setLabel('bottom', 'Pixel level [0-255]')
```

```
    def hist_gauss(self):

        cov = np.reshape(self.gmw2.covariances_, len(self.gmw2.covariances_),1)

        #print(cov)

        means = np.reshape(self.gmw2.means_, len(self.gmw2.means_),1)

        #print(means)

        self.pre_segm(self.img_op)

        v_I_train = (np.reshape(self.imaori,(self.imaori.shape[0]*self.imaori.shape[1],1)))*255 #convertir
a vector

        y,x = np.histogram(v_I_train, bins=256)



        self.histogram_gauss_is_done = 1



        y = y*100/np.sum(y)



        self.hist_lang()

        brush = pg.mkBrush('c')

        self.histogram.plot(x, y, stepMode=True, fillLevel=0, brush=brush)

        self.histogram.addLegend(size = (100,20), offset=(-10,5))




        c                                                                                    =
[(0,0,0),(255,255,0),(255,0,0),(192,192,192),(255,0,128),(255,128,0),(128,0,128),(0,255,0),(0,0,255),(1
66,83,0)]

        for h in range (0,len(cov)):

            mu = means[h]

            variance = cov[h]
```

```
        sigma = math.sqrt(variance)

        x = np.linspace(mu - 3*sigma, mu + 3*sigma, 100)

        u = (scipy.stats.norm.pdf(x, mu, sigma)/scipy.stats.norm.pdf(mu, mu, sigma))*max(y)*self.gmw2.weights_[h]*0.7


        pen = pg.mkPen(color=c[h], width=3)


        self.histogram.plot(x, u, pen=pen, name=("\u03BC :" + str(round(mu,1)) + " ,\u03C3: " + str(round(cov[h],1))))




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


    def prop(self): #propiedades de la segmentada (que era la Otsu)

        self.table_is_done = 1

        self.regions=regionprops(self.label, intensity_image=self.imaori) #en regions guarda las propiedades encontradas por la función regionprops


        self.areas = [prop.area for prop in self.regions] #después va diciendo para cada propiedad a cuál corresponde de las que se han guardado en regions.

        self.centroids=[prop.centroid for prop in self.regions] #lo hace para la característica; etiqueta (qué etiqueta se le ha dado a un área al segmentarla), área, intensidad y centroide.

        self.labels= [prop.label for prop in self.regions]

        self.inte= [prop.mean_intensity for prop in self.regions]


        self.taula.setRowCount(len(self.labels)+1)

        self.taula.setColumnCount(4)
```

```
label_str = []
areas_str = []
centroids_str = []
inte_str = []
for x in range(0,len(self.labels)):
   label_str.append(str(self.labels[x]))
   areas_str.append(str(self.areas[x]))
   centroids_str.append(str(self.centroids[x]))
   inte_str.append(str(self.inte[x]*100))



self.taula.setItem(0,0, QTableWidgetItem(""))
self.taula.item(0, 0).setTextAlignment(QtCore.Qt.AlignCenter)
self.taula.item(0, 0).setBackground(QtGui.QColor(0,183,0))



self.taula.setItem(0,2, QTableWidgetItem("(x , y)"))
self.taula.item(0, 2).setTextAlignment(QtCore.Qt.AlignCenter)
self.taula.item(0, 2).setBackground(QtGui.QColor(0,183,0))

self.taula.setItem(0,3, QTableWidgetItem("e-100"))
self.taula.item(0, 3).setTextAlignment(QtCore.Qt.AlignCenter)
self.taula.item(0, 3).setBackground(QtGui.QColor(0,183,0))
```

```
        self.taula.setEditTriggers(QtGui.QAbstractItemView.NoEditTriggers)

        self.guardar_tabla.setEnabled(True)
        self.action_rea_Intensitat.setEnabled(True)

        for row in range(1,len(label_str)+1):
            self.taula.setItem(row,0, QTableWidgetItem(label_str[row-1]))
            self.taula.item(row, 0).setBackground(QtGui.QColor(237,237,239))
            self.taula.item(row, 0).setTextAlignment(QtCore.Qt.AlignCenter)

            self.taula.setItem(row,1, QTableWidgetItem(areas_str[row-1]))
            self.taula.item(row, 1).setBackground(QtGui.QColor(237,237,239))
            self.taula.item(row, 1).setTextAlignment(QtCore.Qt.AlignCenter)

            self.taula.setItem(row,2, QTableWidgetItem(centroids_str[row-1]))
            self.taula.item(row, 2).setBackground(QtGui.QColor(237,237,239))
            self.taula.item(row, 2).setTextAlignment(QtCore.Qt.AlignCenter)

            self.taula.setItem(row,3, QTableWidgetItem(inte_str[row-1]))
            self.taula.item(row, 3).setBackground(QtGui.QColor(237,237,239))
            self.taula.item(row, 3).setTextAlignment(QtCore.Qt.AlignCenter)

        self.prop_lang()

    def prop_lang(self):
        if self.actionCatal.isChecked():
```

```
        self.taula.setHorizontalHeaderLabels(('Etiqueta','Àrea','Centroide','Intensitat'))
        self.taula.setItem(0,1, QTableWidgetItem("Núm. píxels"))

    elif self.actionCastellano.isChecked():
        self.taula.setHorizontalHeaderLabels(('Etiqueta','Área','Centroide','Intensidad'))
        self.taula.setItem(0,1, QTableWidgetItem("Núm. píxeles"))

    elif self.actionEnglish.isChecked():
        self.taula.setHorizontalHeaderLabels(('Label','Area','Centroid','Intensity'))
        self.taula.setItem(0,1, QTableWidgetItem("Num. pixels"))

    self.taula.item(0, 1).setTextAlignment(QtCore.Qt.AlignCenter)
    self.taula.item(0, 1).setBackground(QtGui.QColor(0,183,0))

def scatter(self):
    self.graph_is_done = 1
    rng = np.random.RandomState(0)

    colors = rng.rand(len(self.areas))

    # plot the data
    self.fig = plt.figure()

    self.ax = self.fig.add_subplot(1, 1, 1)

    if self.actionCatal.isChecked():
```

```
        self.ax.clear()
        self.fig.canvas.set_window_title('Gràfic')
        self.ax.set_title('Àrea / Intensitat', fontsize=20)
        self.ax.set_xlabel('Intensitat', fontsize=15)
        self.ax.set_ylabel('Àrea', fontsize=18)

    elif self.actionCastellano.isChecked():
        self.ax.clear()
        self.fig.canvas.set_window_title('Gráfico')
        self.ax.set_title('Área / Intensidad', fontsize=20)
        self.ax.set_xlabel('Intensidad', fontsize=15)
        self.ax.set_ylabel('Área', fontsize=18)

    elif self.actionEnglish.isChecked():
        self.ax.clear()
        self.fig.canvas.set_window_title('Graph')
        self.ax.set_title('Area / Intensity', fontsize=20)
        self.ax.set_xlabel('Intensity', fontsize=15)
        self.ax.set_ylabel('Area', fontsize=18)

    self.ax.scatter(self.inte, self.areas, c=colors, alpha=0.3,
                    cmap='magma')

    self.fig.patch.set_facecolor(color= '#ededef')

    plt.show()
```

```
def handleSave(self):

    self.list = {'Etiqueta': self.labels,'Area':self.areas,'Centroide': self.centroids, 'Intensitat':self.inte}
    self.data = pd.DataFrame(self.list, columns=['Etiqueta','Area','Centroide','Intensitat'])

    path, ok = QFileDialog.getSaveFileName(
        self, 'Guardar Fitxer', '', 'CSV(*.csv)')

    if ok:
      self.data.to_csv(path, index = None, header=True)
```