Universitat de les Illes Balears

escola politècnica superior

Treball Final de Grau

GRAU D'ENGINYERIA ELECTRÒNICA INDUSTRIAL I AUTOMÀTICA

Exploració de l'ús de càmeres tèrmiques en anàlisi de corrents d'aigua

Treball Final de Grau a l'Escola Politècnica Superior

DANIEL GARCÍA GALMÉS

Tutor
Bartomeu Alorda Ladària

Escola Politècnica Superior
Universitat de les Illes Balears
Palma, setembre de 2017

SUMARI

ÍNDEX DE FIGURES

ÍNDEX DE TAULES

Acrònims

GUI Graphic User Interface

SIG Sistema de Informació Geogràfica

CIR Color InfraRed

TABI Termal Airborne Broadband Imager

LiDAR Light Detection and Ranging

Resum

En aquest treball es proposa el tractament d'imatges tèrmiques com un medi per determinar els fluxos d'aigües pluvials així com les zones de drenatge. Ambdues coses son objectiu principal d'estudi a la hidrologia.

S'ha fet servir un terreny artificial que simula un vessant natural, damunt el qual s'ha vessat aigua per emular la pluja. Les imatges tèrmiques determinen la diferència d'emissió infraroja derivada de la diferència de temperatures de cada part del terreny i de l'aigua a més de la reflectivitat dels materials. Encara que, en aquest cas, el gradient de temperatura no és massa pronunciat, ha de ser possible separar l'aigua de la resta de la imatge mitjançant un algoritme de tractament d'imatges adequat. Aquest estudi pot ser molt útil tant a l'àmbit de la hidrologia com al de l'agricultura.

Termes clau: Fluxos pluvials; processament d'imatges; imatges color infraroig; mapeig drenatge; MATLAB

Abstract

In this paper image processing is proposed as a way to determine rainwater flow and drainage areas. Both targets are primary objectives of hydrology studies.

An artificial terrain that simulates a natural slope is used. Water will be poured on top to emulate rainfall. Infrared color images determine the difference between the infrared emissions at every single point of the terrain resulting from the temperature difference and the reflectivity of the material. Although in this case, the temperature gradient is not too pronounced, it must be possible to differentiate the water from the rest of the image through an appropriate image processing algorithm. This study may be really useful both in hydrology and agriculture.

Keywords: Rain flow mapping; image processing; color infrared image; drainage area mapping; MATLAB

CAPÍTOL 1

ANÀLISI GENERAL DE L'ESTUDI

1.1. Motivació de l'estudi

Aquest projecte surt de l'observació de les grans possibilitats que ofereix l'anàlisi de la fotografia digital degut a l'elevada quantitat d'informació que es pot extreure d'una sola fotografia digital. L'arribada dels avançaments tecnològics de les tècniques d'obtenció (càmeres digitals, *drones* ...) així com de les possibilitats de filtrar la informació que es consideri pertinent per l'estudi, el seu posterior anàlisi i les possibilitats de una automatització de les tècniques (software avançats com per exemple MATLAB) ha obert, ja fa un anys, un camp molt ampli a la investigació per a la aplicació a molts diversos àmbits. S'ha pensat que un d'aquests àmbits possibles és l'estudi de fluxos d'aigües i la seva possible aplicació a l'agricultura.

Descripció del problema

En aquest treball es farà un estudi de recorreguts d'aigua mitjançant la termografia. L'objectiu principal és descobrir si és possible el tractament d'imatges termogràfiques pel seguiment o el desenvolupament dels fluxos d'aigües pluvials i fluvials.

Les imatges tèrmiques mostren la temperatura de cada píxel de la fotografia a partir de colors (groc per a la màxima temperatura i blau fosc per la mínima).

Es sap que l'aigua és més constant tèrmicament que la terra degut a la diferència entre el calor específic dels materials. Mentre que la capacitat calorífica de l'aigua és $4.18\,\mathrm{J}\cdot\mathrm{g}^{-1}\cdot\mathrm{K}^{-1}$, la del sòl és $0.8-0.9\,\mathrm{J}\cdot\mathrm{g}^{-1}\cdot\mathrm{K}^{-1}$, en funció del material que el forma. És a dir, la capacitat calorífica de l'aigua és 4 o 5 vegades superior a la del sòl. Per tant, per a una aportació de calor igual a ambdós, l'aigua eleva la seva temperatura una cinquena part que el que ho fa el sòl. Per exemple, durant el dia, l'arena de la platja té una temperatura superior a l'aigua de la mar, mentre, a la nit, és a la inversa.

Per tant, és quasi impossible que l'aigua i la terra que l'envolta estiguin a la mateixa temperatura, i més si parlem de l'aigua de pluja. Aquest aigua estarà quasi sempre a una temperatura inferior a la de l'entorn, o en dies molt freds, a una temperatura superior.

Aquest fet sembla que permetrà identificar els recorreguts de l'aigua mitjançant la termografia.

Anàlisi d'una imatge de prova

Encara que un estudi complet del problema excedeix els límits d'aquest treball, s'ha cregut convenient aplicar el procés complet a una imatge obtinguda en condicions de laboratori.

Per obtenir les imatges de partida d'aquest estudi, s'empra una càmera termogràfica *FLIR C2*. Les especificacions de la càmera es poden veure a l'annex A.

És important tenir en compte que la reflectivitat infraroja varia segons els materials i el seu estat. En aplicar l'estudi a fotografies preses de terrenys reals, els resultats d'aquest treball tindràn més validesa juntament amb un Sistema de Informació Geogràfica (SIG) que comprengui la composició del terreny, així com la reflectància dels materials i el mapa topogràfic i hidrològic ja que proporcionarà un conjunt més complet de dades.

Evidentment, si els resultats de l'estudi són satisfactoris, caldria un estudi molt més ampli per tal de veure si la tècnica és aplicable a altres imatges similars o, fins i tot, a altres tipus de imatges fent petites modificacions als paràmetres o a l'ordre dels filtres aplicats. També s'ha pensat que seria interessant i possible fer una automatització del procés per tal que es pugui analitzar una gran quantitat d'imatges sense que sigui necessària la intervenció de personal especialitzat i, d'aquesta manera proporcionar una eina útil a altres equips d'investigació.

Encara que s'han fet proves damunt vàries imatges aquest treball s'ha enfocat en mostrar els resultats d'una sola imatge per tal de mostrar el desenvolupament de l'algoritme.

Objectius

El principal objectiu d'aquest estudi és l'anàlisi de la informació que proporcionen els píxels d'una imatge tèrmica i el diseny d'un algoritme de processament d'imatges que apliqui els filtres necessaris per tal d'eliminar tots els elements no importants per la investigació que es vol dur a terme.

Com no és d'interès l'aparença de la imatge sinó el valor representatiu de les temperatures s'emprarà una eina de tractament de dades matemàtiques com és MATLAB. Aquesta eina permet tractar les imatges com matrius de dades bidimensionals, on cada cel·la de la matriu indica un valor numèric relacionat amb la temperatura d'aquest punt de la imatge.

La finalitat és obtenir un algoritme o conjunt d'algoritmes que puguin ser emprats per totes les imatges que es volen tractar, fins a trobar un recorregut d'aigua vàlid i representable damunt la imatge original.

L'altre objectiu és proporcionar per aquest algoritmne una interfície d'usuari fàcil i intuïtiva per l'anàlisi posterior d'altres fotografies.

1.2. Estratègia proposada per resoldre el problema

Es disposa un escenari de laboratori preparat amb un caramull de terra on es vessa aigua per tal de crear un recorregut d'aigua simulant un flux d'aigua real. S'agafen unes

imatges amb la càmera termogràfica. A partir de la imatge tèrmica, s'intentarà preparar un algoritme amb els filtres més adequats de MATLAB per obtenir una imatge final que marqui el recorregut de l'aigua.

Una vegada decidit l'algoritme més adient, es prepararà una interfície d'usuari intuitiva.

1.3. Discusió sobre alternatives d'implementació

La pregunta clau en aquesta discussió és quines són les necessitats en el problema plantejat. L'eina elegida haurà de ser diferent si la resposta és que es vol un programa que apliqui els filtres triats i manipuli les imatges o bé es vol un anàlisi matemàtic de les dades que es poden obtenir d'una imatge digital per després aplicar els filtres més adients a la investigació que es vol dur a terme.

- **Python**

  Python és sobre tot un llenguatge de programació i com a tal les seves possibilitats són molt més amples que les de MATLAB, no sols es pot utilitzar per resoldre problemes matemàtics o d'enginyeria sino també a creació de jocs, desenvolupament web... *Python* es flexible, versàtil i multiplataforma. Últimament també es pot disposar de molts paquets ja preparats per altres usuaris que van posant a l'abast públic. Entre ells es poden trobar paquets per filtrat d'imatges, etc.

  També té les avantages de tenir una llicència gratuïta que el fa molt aplicable a estudis petits o a industries modestes.

- **MATLAB**

  MATLAB és principalment una eina matemàtica molt potent que treballa sobre tot amb l'anàlisi de matrius de dades. És una plataforma optimitzada per a la resolució de problemes científics i d'enginyeria. Disposa de *toolbox* específiques de tractament d'imatges a més de una aplicació per poder desenvolupar una interfície adaptada a les necessitats de l'usuari

  Malgrat l'inconvenient que suposa el fet de que utilitza un llenguatge de programació propi i de que la llicència no és gratuïta, les seves prestacions fan de MATLAB l'enina ideal per l'objectiu d'aquest estudi.

CAPÍTOL 2

PRECEDENTS I FONAMENTS

2.1. Precedents

Altres estudis han cercat objectius similars als d'aquest treball per mètodes semblants. Aquests estudis empren també la radiació no visible per determinar els recorreguts d'aigua i, inclús, van més enllà.

Diversos estudis tracten amb radiació UV per determinar els recorreguts d'aigües tèrboles i, a més aconsegueixen determinar la velocitat de l'aigua i les zones de drenatge. Exemples d'aquests treballs son: *Fluorescent particle tracers in surface hydrology: a proof of concept in a semi-natural hillslope* [1] juntament amb el seu treball anterior *Characterization of Buoyant Fluorescent Particles for Field Observations of Water Flows* [2].

En aquests estudis, es treballa la possibilitat de determinar els fluxos d'aigua mitjançant el seguiment d'unes micropartícules fluorescents (biodegradables) susceptibles a la radiació UV. Al treball també s'estudia la possibilitat de determinar la velocitat dels fluxos i els temps de viatge de les partícules.

Les micropartícules tenen un tamany aproximat entre 710 i 1180 μm, son de forma esfèrica i la seva densitat és de 0.98 g/cm^3.

Els instruments emprats per aquest tipus de treball son càmeres amb filtres òptics per radiació UV, juntament amb un sistema simulador de pluja sobre un terreny semi-natural muntat per ells, simulant un vessant natural.

El treball *Mapping subsurface drainage systems with color infrared aerial photographs* [3] proposa un mètode d'imatges tèrmiques aèries (Color InfraRed (CIR)), juntament amb un sistema d'informació geogràfica (SIG), per l'estudi del sistema de drenatge dels terrenys agrícoles de Vermillion County, Illinois (Estats Units).

La finalitat principal és determinar l'estat de les zones de drenatge, ja que la comprovació manual de totes les zones es llarga i complicada. Les zones on el sistema de drenatge funciona de forma eficient s'assequen més ràpid que les zones on hi pot haver un problema a la línia de drenatge i això fa possible el seu estudi.

Un altre article que tracta un tema molt similar és: *Using airborne thermal imaging data to measure near-surface hydrology in upland ecosystems* [4].

En aquest treball es junta la funció del mapeig d'imatges tèrmiques aèries (Termal Airborne Broadband Imager (TABI)) amb la mesura topogràfica del terreny mitjançant tecnologia Light Detection and Ranging (LiDAR). A més emprant un SIG de dades existents del terreny i la flora, es disposa d'una gran quantitat d'informació (com l'emissivitat dels distints materials i de la vegetació del terreny i la ubicació dels sistemes de drenatge antropogènics).

L'objectiu principal d'aquest treball és determinar si es pot identificar les zones de drenatge antropogèniques mitjançant TABI i LiDAR després del canvi hidrològic ocasionat als pantans del Regne Unit per la intensificació de la agricultura i l'extracció de torba (un carbó natural lleuger i molt ric en nitrogen emprat majoritàriament per fer abonament).

2.2. Fonaments

Els fonaments d'aquest treball son bàsicament l'exploració matemàtica de les imatges tèrmiques. S'empra una eina matemàtica pel tractament d'imatges com a matrius de valors numèrics.

En aquesta secció es tractan els fonaments del tractament digital d'imatges i les tècniques i funcions emprades per aconseguir l'algoritme en que es basa aquest treball.

2.2.1. Processament digital d'imatges

És el conjunt de tècniques que s'apliquen a les imatges digitals amb l'objectiu de millorar la qualitat o facilitar la cerca d'informació'. [5]

A aquest estudi la referència agafada és la de facilitar la cerca d'informació on s'hi afegeix l'objectiu de remarcar certs valors damunt la imatge original. Això es farà emprant l'eina MATLAB, aplicant funions d'aquest programa damunt les imatges.

S'ha d'entendre la imatge com una matriu de valors numèrics, on cada píxel és una cel·la de la matriu que aporta un valor del color representat.

Com es disposa d'una imatge en color s'ha d'entendre la imatge com un conjunt de tres matrius de dades. Una per cada color primari que conforma el registre de colors RGB (vermell, verd i blau). Per tant la imatge és una matriu tridimensional de dades numèriques.

A les imatges RGB emprades a la fotografia digital, les tres matrius que representen els colors tenen la mateixa mida, i cada una té un valor entre 0 i 255 a cada cel·la que indica la quantitat de vermell, verd i blau que disposa aquest píxel per reproduir el seu color. A partir d'aquí es conformen la resta de colors (fins a 256^3 = 16.8 milions de colors). És a dir, a cada punt de la imatge li correspon un vector tridimensional de components entre 0 i 255. La manca dels tres colors conforma el color negre, mentre que la suma del màxim dels tres colors representa el color blanc, com es mostra a les figures 2.1 i 2.2.

Cal dir que aquesta tècnica de suma de colors, només és una de les tècniques emprades per la conformació de colors. Es coneix com síntesi additiva de colors RGB, on s'empra llum dels tres colors primaris sobre un fons absent de color (negre) per

conformar els colors. És també el cas de síntesi de colors emprat en els monitors d'ordinador i, per tant, serà la tècnica emprada al treball.

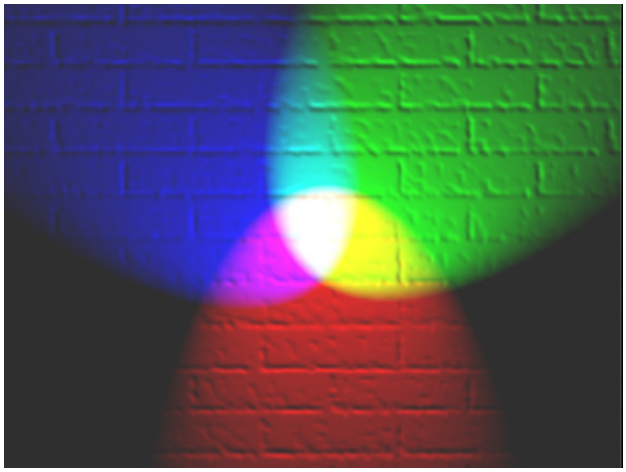

Figura 2.1: Síntesi additiva de color RGB: Llums dels colors primaris sumant-se en el colors secundaris i blanc. Font: https://es.wikipedia.org

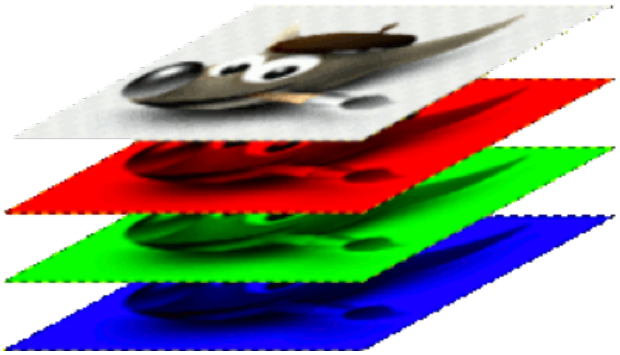

Figura 2.2: Síntesi additiva de color RGB: Capes dels colors primaris que conformen la imatge final. Font: https://docs.gimp.org

En cas d'impressió s'empra la tècnica contraria a aquesta. La síntesi sostractiva de colors CYMK. En aquesta tècnica es resten els colors primaris (cien, groc i magenta) del fons blanc del paper. Un esquema de les dues tècniques es pot veure a la figura 2.3.

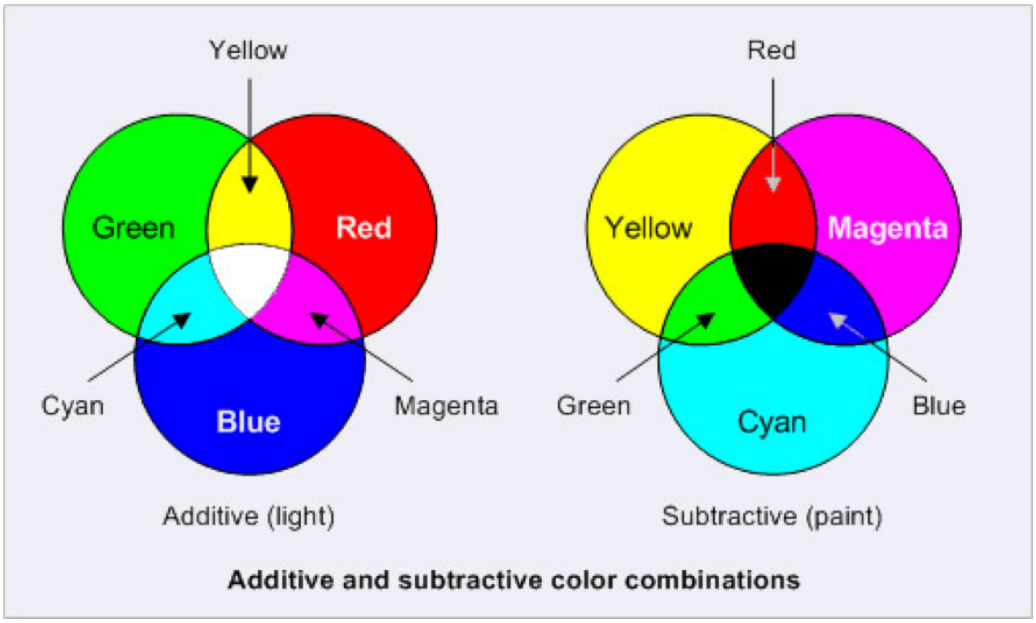

Figura 2.3: Comparativa síntesi additiva i síntesi sostractiva. Font: https://soma.sbcc.edu

Les imatges en escala de grisos son una subclasse de les imatges d'intensitat (veure pàgina 54 de l'apèndix C). A cada punt de la imatge se li associa un valor decimal (`double`) comprés entre 0 i 1.

Les imatges binaritzades en blanc i negre associen a cada punt el valor 0 pel negre i 1 pel blanc.

2.2.2. Filtrat

En aquest apartat s'explica en termes generals en qué consisteix un filtrat d'imatge i els tipus de filtrats existents, ja que la part principal d'aquest estudi consisteix en un filtrat d'una imatge tèrmica per tal d'extreure i resaltar la part que correspon a un flux d'aigua sobre un terreny.

El procés de filtrat digital té com objectiu modificar la imatge original fins obtenir una imatge final amb unes característiques més adequades per una aplicació específica, recalcant certes propietats que possibilitin efectuar operacions de processat damunt ella.

Els filtres emprats per el processat d'imatges digital es diferencien en filtres en el domini de la freqüència i filtres en el domini de l'espai, que s'expliquen a continuació.

Filtrat en el domini de la freqüència

Els filtres de freqüència treballen damunt el domini de la freqüència així com ho farien els filtres electrònics. Per aconseguir treballar en domini freqüencial damunt una imatge digital, es necessita treballar damunt la transformada discreta de Fourier en dues dimensions (Eq. 2.1 i Eq. 2.2), aplicant el teorema de convolució circular (Eq. 2.3).

$$F(u,v) = \mathscr{F}\{f(x,y)\} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})} \tag{2.1}$$

Eq. 2.1. Transformada discreta de Fourier per un espai bidimensional.

$$f(x,y) = \mathscr{F}^{-1}\{F(u,v)\} = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(\frac{ux}{M}+\frac{vy}{N})} \tag{2.2}$$

Eq. 2.2. Transformada inversa de Fourier per un espai bidimensional.

on M i N son les dimensions de la imatge a tractar, u i v les variables de freqüència corresponents a x i y del domini espacial i j la unitat imaginaria.

El teorema de convolució circular indica que una convolució de dues seqüències infinites es pot obtenir com la transformada inversa del producte de les transformades de ambdues seqüències. Per tant:

$$g(x,y) = \mathscr{F}^{-1}\{\mathscr{F}\{f(x,y)\} \cdot H(u,v)\} \tag{2.3}$$

Eq. 2.3.Teorema de convolució circular.

on f és el senyal discret inicial a l'espai x,y i $H(u,v)$ el filtre atenuador de freqüències.

Com la teoria de la convolució és idèntica a la pròpia en el domini de l'espai, es podria afirmar que tots aquests filtres poden implementar en un filtre espacial. A la figura 2.4 es mostra un esquema del procés que es faria servir en cas d'aplicar aquest tipus de filtre.

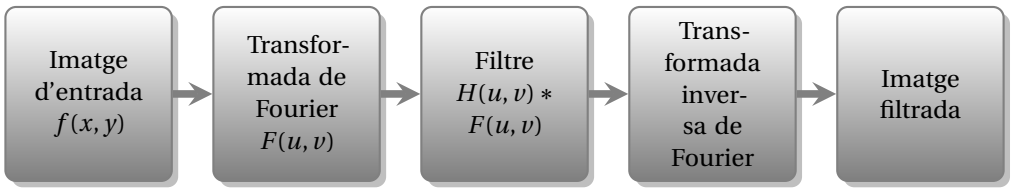

Figura 2.4: Etapes del filtrat en domini freqüencial. (Font: https://soma.sbcc.edu)

Filtrat en el domini de l'espai

Degut a que l'eina emprada per el tractament d'imatges és MATLAB, que es basa en el tractament de matrius, aquest tipus de filtrat és el mètode que s'utilitza per realitzar l'estudi d'aquest treball.

El domini de l'espai fa referència al propi plànol de la imatge. Es treballa directament damunt els píxels de la imatge. Els processos duts a terme en aquest domini es definen damunt el propi píxel a estudiar, juntament amb un veïnat de píxels circumdants. Aquest conjunt de píxels és normalment un quadrat amb el píxel objectiu d'estudi com a centre.

Per a processar la imatge completa aquesta regió quadrada (màscara o finestra) es mou píxel a píxel començant des de l'origen de la imatge (el vèrtex de dalt a l'esquerra). Es pot veure un esquema de la màscara a la figura 2.5.

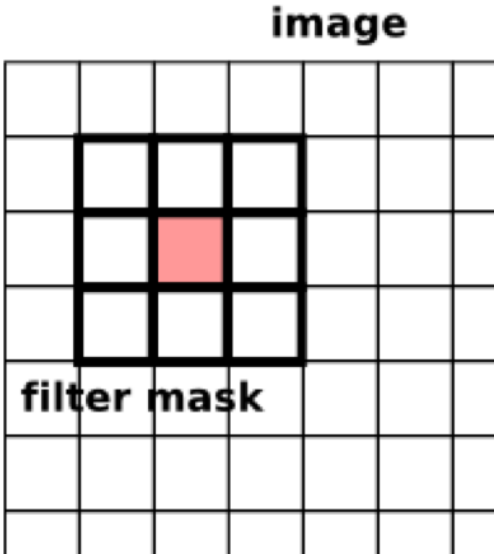

Figura 2.5: Màscara de treball 3x3 damunt imatge. Font: http://nullprogram.com

El problema d'aquesta tècnica és a l'hora d'agafar una màscara de píxels de les vores de la imatge. Com es pot veure a la figura 2.5, quan l'objectiu d'estudi és el píxel inicial (posició 1,1), la màscara només disposaria dels 3 píxels de baix a la dreta. Per aquesta raó a l'hora de processar una imatge en domini de l'espai s'ha d'expandir la imatge en totes direccions. En el cas de la figura 2.5 s'ha d'afegir un píxel més en cada direcció i omplir-lo amb valors que ens permetin estudiar els píxels de les vores. Aquest procés d'expansió de la imatge amb valors adequats s'anomena *padding*.

Els valors triats per omplir aquestes noves cel·les de la imatge solen ser zeros o valors simètrics als de la imatge, és a dir, com si es mirés un mirall a les vores de la imatge. Podem veure les dues tècniques a les figures 2.6 i 2.7. A la figura 2.7 s'empra l'opció de *padding 'replicate'*, que per el cas d'expansió per un filtre 3x3 (1 píxel) és equivalent a l'opció *'symmetric'*. Les cel·les blaves són les afegides per al processament de la imatge.

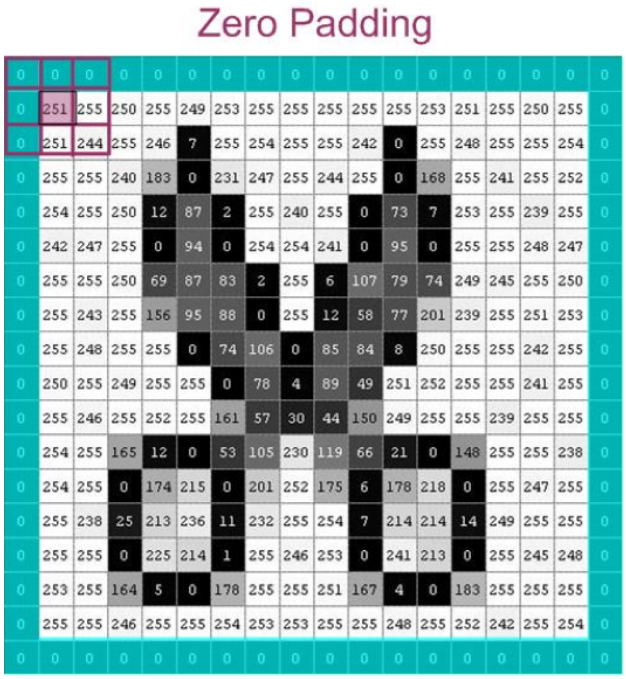

Figura 2.6: *Padding* emprant zeros. (Font: http://www.cs.uregina.ca)

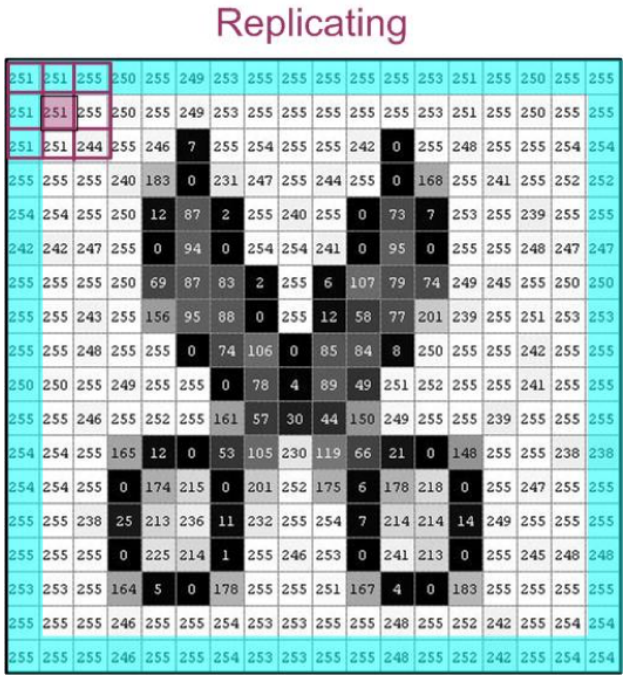

Figura 2.7: *Padding* simètric. (Font: http://www.cs.uregina.ca)

El domini espacial disposa de dos mètodes de filtrat principals. Aquests mètodes es corresponen amb operacions lineals i no lineals per el càlcul del nou valor de la cel·la on s'està aplicant el filtre:

- **Filtrat espacial lineal**
  Aquest tipus de filtrat consisteix en multiplicar cada píxel de la màscara (píxel d'interès més veïnat) per els coeficients corresponents al filtre i sumar els valors resultants en el nou valor que substituirà al de la cel·la d'estudi.

- **Filtrat espacial no lineal**
  Aquest segon tipus consisteix en operacions no lineals a l'hora de triar un nou valor per el píxel objectiu. Algunes d'aquestes operacions podrien ser triar el màxim valor o la mediana de la secció d'estudi.

Després de provar els dos mètodes, s'ha decidit emprar el filtrat no lineal ja que han aportat una major definició dels resultats.

2.2.3. MATLAB

MATLAB és una eina molt potent de software matemàtic que ofereix un entorn de desenvolupament integrat amb un llenguatge de programació propi. El llenguatge de MATLAB és un llenguatge d'alt rendiment per computació tècnica, que integra computació, visualització i programació.

Cal destacar entre les prestacions bàsiques de MATLAB la manipulació de matrius, la representació de dades i funcions, la creació d'algoritmes, la creació d'interfícies d'usuari (GUI) i la comunicació amb programes d'altres llenguatges i amb dispositius hardware.

MATLAB té un gran potencial de tractament de dades en forma de matrius en tots els àmbits aplicables. Com s'ha explicat, interessa emprar el filtrat espacial d'imatges al tractar-se d'imatges RGB en format digital, per tant és una eina que s'adapta perfectament a les necessitats de l'estudi.

Les prestacions de MATLAB es divideixen en caixes d'eines o *toolboxes*. La que s'empra per processament digital es diu *Image Processing Toolbox* i ve preinstal·lada amb el paquet de MATLAB.

Aquesta *toolbox* permet tant llegir i escriure arxius amb diferents tipus d'extensió de tipus imatge, definint-los dins el programa com a matrius de dades, com modificar les pròpies imatges mitjançant funcions de millora, modificació, processat o filtrat i és la més adequada per aquest treball.

CAPÍTOL 3

IMPLEMENTACIÓ DE L'ALGORITME PROPOSAT

L'algoritme de processat utilitzat a aquest treball està format per unes funcions determinades del *Image Processing Toolbox de* MATLAB a més d'unes funcions d'elaboració pròpia. Es descriuen en primer lloc aquestes funcions pròpies per poder utilitzar-les després a l'algoritme principàl proposat en aquest treball per l'estudi de les imatges.

3.1. Funcions pròpies

Són unes funcions que s'han implementat als algoritmes de processat de les imatges per tal d'aconseguir l'objectiu del treball i facilitar el seu enteniment.

3.1.1. Funció `RGBmedfilt`

Aquesta funció no és més que una ampliació de `medfilt2` (explicat a la pàgina 61 de l'apèndix C) del *Image Processing Toolbox de* MATLAB. Per facilitar la lectura dels algoritmes s'ha adaptat la funció per poder emprar aquest filtre amb imatges RGB.

S'ha implementat la funció intentant no alterar la funcionalitat i el mètode de cridada de la funció original. La única diferència és la parametrització de la mida de la màscara del filtre. En aquest cas s'ha implementat de forma que només sigui possible emprar una màscara quadrada i, per tant, només s'introdueix un valor numèric que senyala el valor del costat del quadrat.

```
49      %Inici
50
51 -    g_R = g(:,:,1);
52 -    g_G = g(:,:,2);
53 -    g_B = g(:,:,3);
54
55 -    f_R = medfilt2(g_R, s, [W W]);
56 -    f_G = medfilt2(g_G, s, [W W]);
57 -    f_B = medfilt2(g_B, s, [W W]);
58
59 -    f(:,:,1) = f_R;
60 -    f(:,:,2) = f_G;
61 -    f(:,:,3) = f_B;
62
```

Figura 3.1: Cos de la funció `RGBmedfilt`.

El tipus de parametrització `varargin` permet no tindre que emplenar tots els paràmetres d'entrada de la funció. D'aquesta forma s'empran els valors per defecte de la funció original en cas de no escriure algun dels paràmetres possibles.

La funció, bàsicament, divideix la imatge de color real en tres imatges d'intensitat corresponents a les tres capes de color. Llavors, filtra cada una de les capes per separat amb la funció original `medfilt2` i torna a ajuntar les 3 capes en una sola imatge final de tipus RGB.

```
1       function f = RGBmedfilt(varargin)
2       %RGBMEDFILT(G, S, W) La imatge RGB G es passa per un filtre de medianes
3       %amb opcions de padding s i tamany W. En cas de no introduir valors per s
4       %i W s'empraràn valors 'symmetric' i '3x3' respectivament.
5       %Per ajuda de medfilt2 escriure 'help medfilt2'
6
7 -     g = varargin{1};
8
9       %Comprobació inicial
10
11 -    d = size(g);
12 -    d = [d 0 0];
13 -    if(~strcmp(class(g), 'uint8') && ~strcmp(class(g), 'double') || d(1,3) ~= 3)
14 -        error('Incorrect image "g" format.')
15 -    end
16
```

Figura 3.2: Inicialització de la funció. Comprovació del tipus de matriu d'entrada i de les dimensions.

Igualment, s'ha dut a terme una comprovació del tipus d'imatge i el tractament d'errors dels paràmetres d'entrada corresponent per tal d'aconseguir una funció que pot ser utilitzada per diferents àmbits i per altres usuaris que no coneixen la funció.

També disposa d'un text d'ajuda sobre com emprar la funció que es pot llegir amb la comanda `help RGBmedfilt` de la mateixa manera que per les altres funcions que ens ofereix MATLAB.

Es pot veure la funció completa a l'apèndix E.

3.1.2. Funció `complementar`

És una funció senzilla que agafa d'entrada una imatge de intensitat en escala de grisos o una imatge binària i retorna la imatge complementària a aquesta imatge.

Després d'una comprovació del tipus d'imatge i dels valors que la composen, un doble bucle `for` s'encarrega de complementar els valors.

Finalment, si els valors de la imatge s'han convertit a `double` per ser processats (imatge en escala de grisos) es torna a convertir la imatge al tipus `uint8` d'entrada.

```
22      %Inici
23
24 -    [M, N] = size(g);
25
26 -    for i = 1:1:M
27 -        for j = 1:1:N
28 -            f(i,j) = 1 - g(i,j);
29 -        end
30 -    end
31
```

Figura 3.3: Cos de la funció complementar.

Es pot veure la funció completa a l'apèndix F.

3.1.3. Funció `remarcar`

Aquesta funció es basa en ajuntar dues imatges en una. Introduint per paràmetre una imatge RGB i una imatge binària, els valors alts (iguals a 1) de la imatge binària es marcaran a la imatge RGB del color que s'indiqui en codi de color RGB.

El codi de colors RGB és simple: la combinació de tres valors determinen la composició de tots els colors possibles a la imatge. El primer valor correspon a la capa vermella, el segon a la verda i el tercer a la blava. En cas de voler pintar de color vermell, s'empren els valors [255, 0, 0], pel verd, [0, 255, 0] i pel blau, [0, 0, 255].

```
1    function f = remarcar(g, h, R, G, B)
2    %REMARCAR Marca los valores altos de una imagen en otra.
3     %f = remarcar(g, h, R, G, B)
4     %   Remarca los valores altos(1s) de una imagen "h" en la imagen "g"
5     %   del color definido por los valores R, G y B, correspondientes
6     %   al código de colores RGB del color deseado.
7     %   h - imagen binaria | g/f - imagen RGB | R/G/B - uint8 [0-255]
8     %   Ejemplos R/G/B: 255, 0, 0 - Rojo | 0, 255, 0 - Verde |
9     %   0, 0, 255 - Azul | 255, 255, 255 - Blanco | 0, 0, 0 - Negro
10
11    %Comprobación inicial
12
13 -  if (R < 0 || G < 0 || B < 0 || R > 255 || G > 255 || B > 255)
14 -      error('R/G/B values must be unsigned 8-bit integers [0-255].')
15 -  end
16
17 -  d = size(g);
18 -  d = [d 0];
19 -  if (~strcmp(class(g), 'uint8') && ~strcmp(class(g), 'double') || d(1,3) ~= 3)
20 -      error('Incorrect image "g" format.')
21 -  end
```

Figura 3.4: Inicialització i part de les comprovacions d'errors de parametrització de la funció remarcar.

Després de vàries comprovacions de tipus d'imatges d'entrada i dels valors del codi de color RGB, es disposa la imatge RGB en les diferents capes i es pinten els valors dels píxels de cada capa que es corresponguin amb un 1 de la imatge binària amb els valors del codi RGB indicat per l'usuari.

Un doble `for` i un condicionant `if` a dins el segon `for` implementen aquesta funció.

```
56 -  for i = 1:1:M
57 -      for j = 1:1:N
58 -          if (h(i,j) == 1)
59 -              f_R(i,j) = R;
60 -              f_G(i,j) = G;
61 -              f_B(i,j) = B;
62 -          end
63 -      end
64 -  end
```

Figura 3.5: Cos de la funció remarcar.

Es pot veure el codi complet de la funció a l'apèndix G

S'ha dut a terme un ampli control d'errors i parametrització per tal de poder ser emprada per altres finalitats i per usuaris que no coneguin la funció. També disposa d'una funció d'ajuda de la mateixa manera que la funció `RGBmedfilt` comentada i explicada als apartats anteriors.

3.2. Algoritme de processat principal

L'algoritme escollit per la part del processament de les imatges tèrmiques es composa de diferents passes. Es pot veure esquematitzat a la figura 3.6

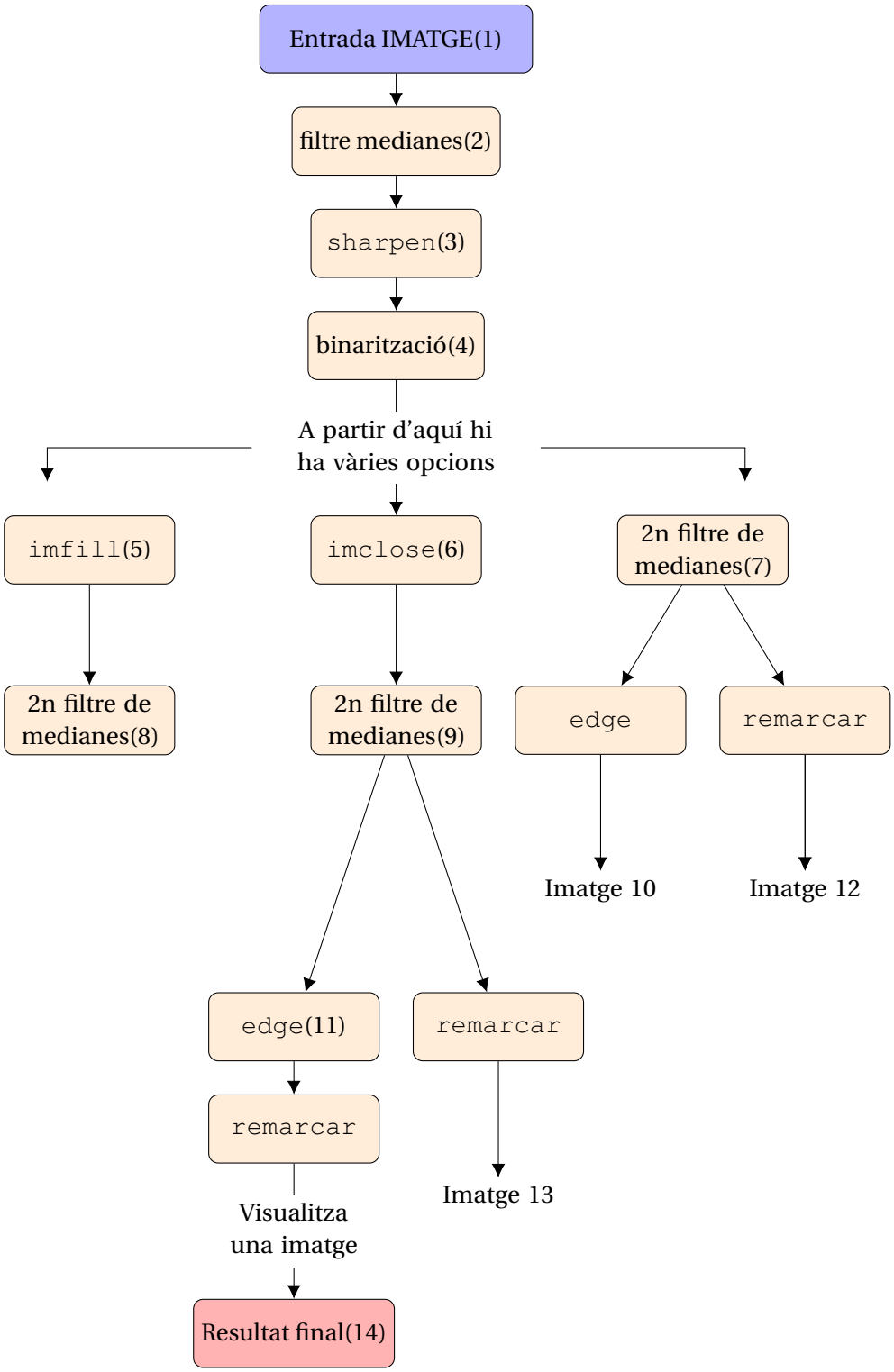

Figura 3.6: Esquema de funcionament de l'algoritme. Els números fan referència a la numeració de les imatges a la figura J.1

Inicialització dels paràmetres

Ajustem els paràmetres que emprarem al desenvolupament de l'algoritme. La mida de la màscara dels filtres, juntament amb els paràmetres `radius` (radi) i `ammount` (quantitat) de la funció `sharpen` i el `threshold` de la binarització.

A la figura 3.7 es veuen les inicialitzacións dels valors de mida de les màscares (`m1` i `m2`), el radi (`r`), el valor *ammount* (`a`) i el de *threshold* (`t`).

La lectura de la imatge de la línea 7 de la mateixa figura es considera també com inicialització ja que serà la imatge emprada a tot l'algoritme i pot ser canviada fàcilment modificant el nom de la imatge.

Aquests valors, estan inicialitzats al principi per tal de poder canviar-los fàcil i ràpidament.

```
1 -    m1 = 9;
2 -    r = 10;
3 -    a = 5;
4 -    t = 0.25;
5 -    m2 = 5;
6
7 -    f = imread('FLIR0094.jpg');
```

Figura 3.7: Inicialitzacions de l'algoritme

Nota: per escollir una imatge que no es troba dins el directori principal de MATLAB, s'ha de definir tota la ruta del directori de la imatge (per exemple:'Users/Dani/Desktop/Flir0094.jp') o canviar el directori de treball amb la comanda cd (Per exemple: 'cd(/Users/Dani/Desktop);')

Filtre de mediana a imatge en format RGB

El filtre escollit per llevar el renou és un filtre de medianes. Aquest filtre, com el seu nom indica, agafa com a valor pel píxel d'estudi el valor que representa la mediana dels valors continguts a la imatge corresponents a la màscara del filtre.

Mitjançant vàries proves amb diversos valors, s'ha escollit una màscara de 9x9 per tal de llevar tot el renou. Com efecte secundari obtenim una imatge d'aspecte desenfocat. No és molt preocupant, ja que el punt d'interés en aquest treball és analitzar el recorregut de l'aigua i queda perfectament visible a pesar de la imatge desenfocada.

S'ha fet servir l'opció de `padding` *'symmetric'*, ja que la imatge que representa un flux d'aigua és una imatge que segueix en totes direccions.

Les altres opcions de `padding` explicades al capítol 2 no s'ajusten a les necessitats d'aquest tipus d'imatge, ja que un flux d'aigua no es comporta d'una forma cíclica i tampoc és una imatge tancada.

Donat això s'ha interpretat que la opció simétrica era la més indicada per emular la part anterior i posterior del flux per simular una imatge més gran.

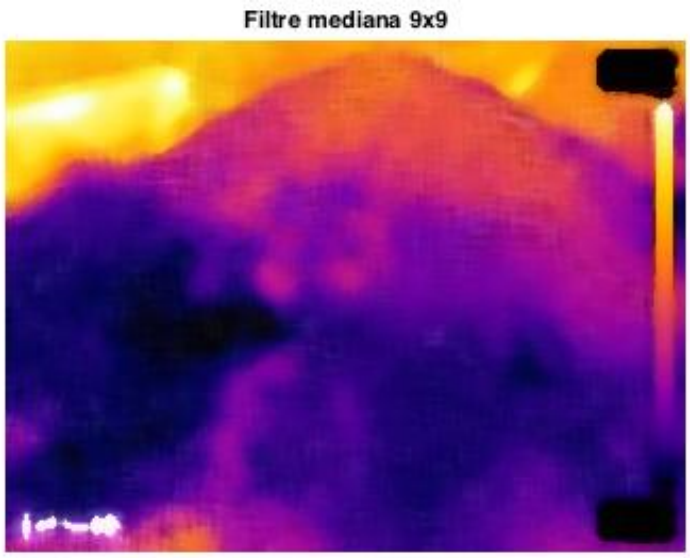

Figura 3.8: Imatge resultant d'aplicar filtre de medianes 9x9

Funció `sharpen`

La funció `sharpen` (esmolar) representa una funció d'enfocament. A part de compensar una mica la imatge desenfocada que ens retorna la funció del filtre de medianes, ens ajuda a realçar les vores del relleu i dels diversos colors.

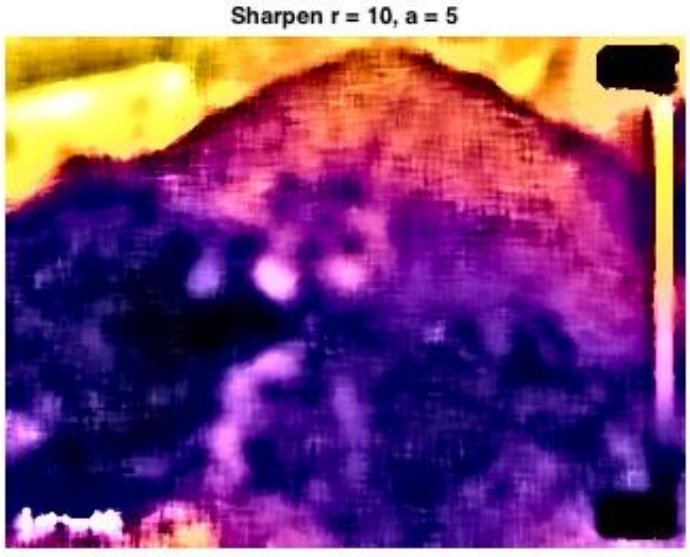

Figura 3.9: Imatge resultant d'aplicar filtre `sharpen`

Els valors han estat triats manualment executant l'algoritme diverses vegades amb valors diferents. Després de realitzar vàries proves amb diversos valors s'ha decidit que els valors elevats obtenen els millors resultats a aquest tipus d'imatge.

Binarització

Amb la binarització s'asignen uns punts com a recorregut d'aigua i altres no, segons la intensitat de cada punt de la imatge.

Es tria com a llindar un valor entre 0 i 1. A partir de la imatge en escala de grisos, cada punt de valor inferior al llindar es transforme a 0 (negre) i cada punt de valor superior al llindar es transforme a 1 (blanc). D'aquesta manera s'aconsegueix una imatge binària (en blanc i negre) que representa el recorregut d'aigua com a blanc.

Es pot fer de forma que es determini el llindar automàticament amb la funció `greythresh`. Si els resultats no son determinants, al menys ja es té un valor de `threshold` per el que començar a pujar o baixar segons les circumstàncies.

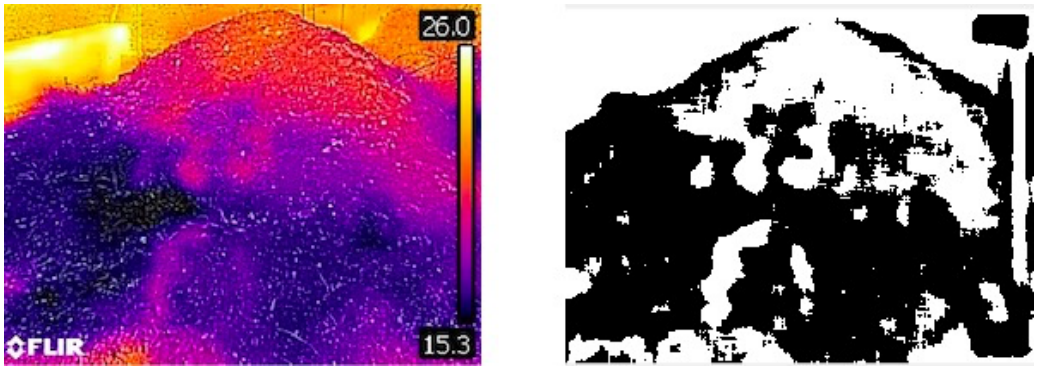

Figura 3.10: Imatge original i després de la binarització.

```
21 -    g = RGBmedfilt2(f, m1);
22
23 -    h = imsharpen(g, 'radius', r, 'amount', a);
24
25 -    hGray = rgb2gray(h);
26
27      %t = graythresh(hGray)
28
29 -    hBin = im2bw(hGray,t);
```

Figura 3.11: Part de l'algoritme corresponent a Filtre medianes, `sharpen` i binarització

Funció `imfill`

Es tracta d'un tipus de funció d'ompliment de la imatge. A una imatge binària, s'intenten omplir els forats que té la part blanca (de valor alt).

Després de binaritzar la imatge el que ens interessa és llevar punts atípics del flux d'aigua i suavitzar les vores del recorregut per acostar la imatge resultant a un flux més real. Per tant es decideix implementar la funció `imfill` amb la imatge invertida per

tal de tancar els forats de la part negre (de valor baix). Després es torna a invertir la imatge.

Encara que sí ens serveix per llevar aquests punts atípics, també lleva algunes parts importants del recorregut de l'aigua.

A l'algoritme es deixa aquesta part per tal de fer comparacions amb l'altre tipus d'ompliment (funció `imclose`), que com es vorà, dona millors resultats.

L'operador de MATLAB emprat normalment per invertir imatges (operador ~) no ha funcionat correctament. Per això, s'ha implementat la funció d'elaboració pròpia `complementar`, explicada a la pàgina14.

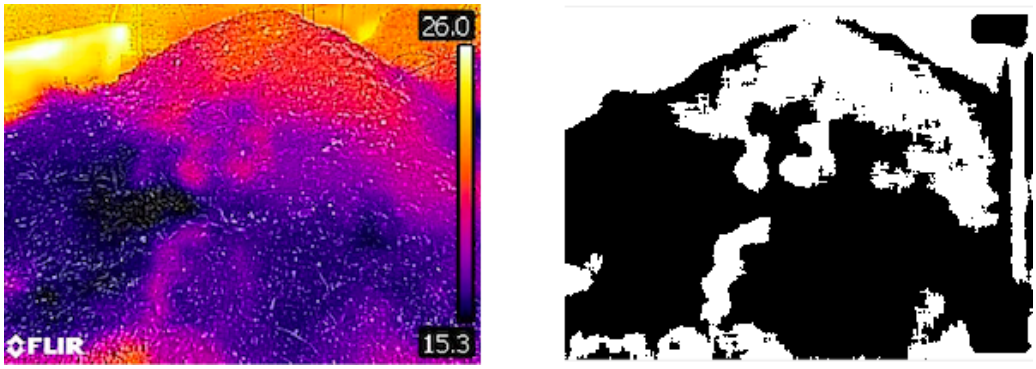

Figura 3.12: Imatges original i després de `imfill`

Funció `imclose`

Es tracta d'un altre tipus de funció d'ompliment de la imatge. Funciona de forma similar a la funció `imfill` però a aquesta funció es pot triar una figura, així com les seves dimensions, amb la que es recorre la imatge per tal d'omplir els forats i suavitzar les vores.

A aquesta funció s'ha triat una figura que té forma de disc amb radi 3. Encara que no lleva totes les zones separades del flux reconegudes com a tal, sí que ens serveix per suavitzar les vores i aconseguir un flux més continu.

La funció s'empra de la mateixa forma que la funció `imfill`, invertint la imatge abans de passar-la per la funció.

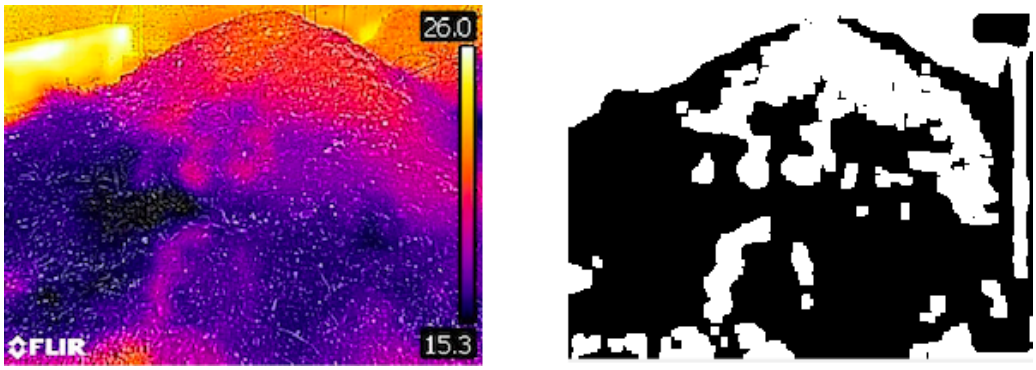

Figura 3.13: Imatges original i després de `imclose`

```
31 -    hBinFill = complementar(hBin);
32
33 -    hBinFill = imfill(hBinFill);
34
35 -    hBinFill = complementar(hBinFill);
36
37 -    iBin = imclose(~hBin, strel('disk', 3));
38 -    iBin = ~iBin;
```

Figura 3.14: Funcions `imclose` i `imfill` (juntament amb la funció implementada per invertir: `complementar`)

Segon filtre de mediana

Es pot aplicar a continuació un altre filtre de medianes per eliminar el renou generat per la funció `sharpen` i la funció de binarització. Aquest filtre es pot aplicar amb una màscara més petita (5x5) ja que es fa sobre una imatge que ja no té el renou de la imatge principal.

Com la imatge filtrada és una imatge binària, no es desenfoca la imatge. I per la mateixa raó, es pot emprar la funció de MATLAB per filtres de medianes (`medfilt2`), que funciona per imatges d'intensitat (escala de grisos o blanc i negre) sense problemes.

Figura 3.15: Imatge resultant de passar el segon filtre de medianes (dreta) a la imatge amb filtre`imclose` (esquerra).

Funció edge

Aquesta funció retorna el perfilat de les formes de la imatge binària. Pot ser una gran eina per mostrar damunt la imatge inicial el que es considera flux d'aigua ja que permet deixar visible la imatge principal.

És una funció que opera sobre imatges en escala de grisos i, el més interessant aquí, també sobre imatges binàries. S'introdueix per paràmetre el nom de l'algoritme emprat per la definició de les vores dels resultats obtinguts. En aquest estudi, s'empra el model *Canny*.

A la imatge 3.16 es pot observar com dibuixa les vores de la forma resultant perfectament, quedant a punt per remarcar la imatge original.

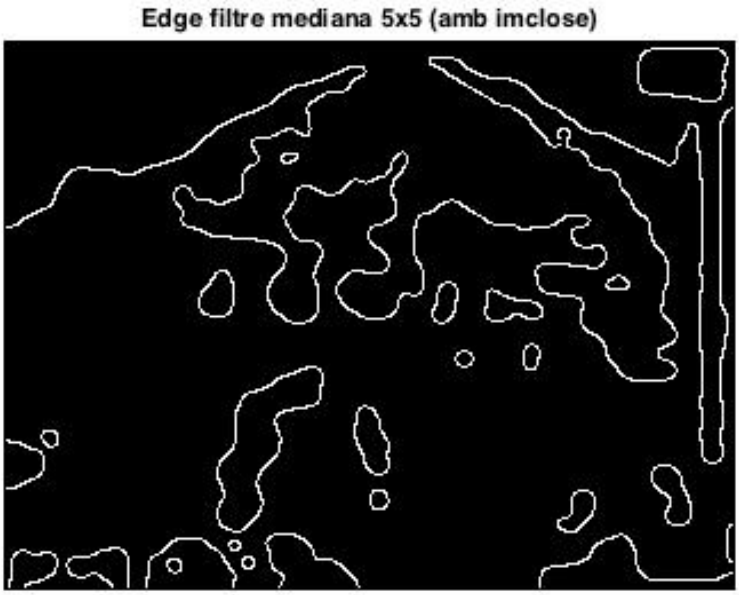

Figura 3.16: Imatge binaritzada amb funció `edge`. `imclose` + filtre medianes + `edge`

Funció `remarcar`

Aquesta funció es pot aplicar es cas de voler destacar visualment amb color el resultat. És una funció d'elaboració pròpia, explicada a la pàgina 15, que utilitza dues imatges (la original i la imatge binària que es vol pintar a sobre) i els valors RGB dels color en que es vol pintar. Retorna una nova imatge amb la imatge binària pintada del color desitjat sobre la imatge que s'introdueix.

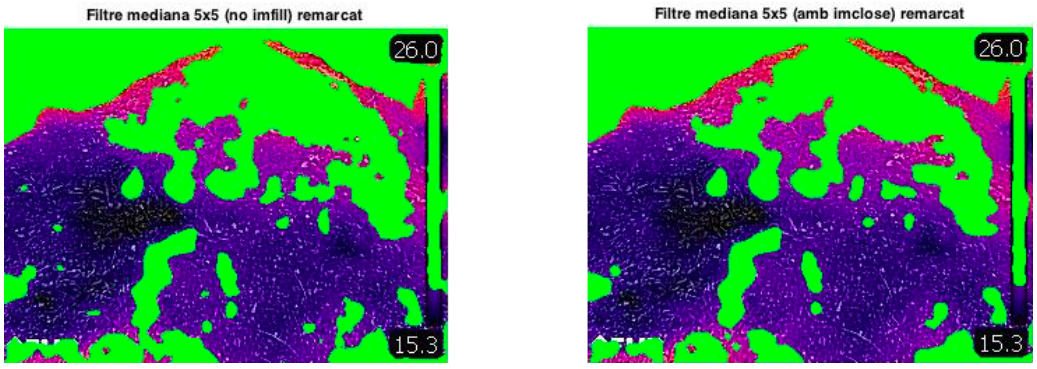

Figura 3.17: Imatges tratades amb la funció `remarcar`. A l'esquerra sense filtre d'ompliment i a la dreta amb `imclose`

En aquest cas es pinta de color verd (per no tindre conflictes amb els colors associats a la temperatura) els resultats de diversos casos de imatges binàries, amb i sense la

funció `edge`, damunt la imatge tèrmica corresponent fent molt més visible el recorregut de l'aigua.

També seria interessant pintar-la damunt la imatge real (no tèrmica) corresponent si es disposa d'ella.

```
44 -    iBinMed = medfilt2(iBin, 'symmetric', [m2 m2]);
45
46 -    hBinEdge = edge(hBinMed, 'canny');
47
48 -    iBinEdge = edge(iBinMed, 'canny');
49
50 -    j = remarcar(f, hBinMed, 0, 255, 0);
51
52 -    k = remarcar(f, iBinMed, 0, 255, 0);
53
54 -    l = remarcar(f, iBinEdge, 0, 255, 0);
```

Figura 3.18: Segon filtre de medianes, funció `edge` i funció `remarcar`

Representació dels resultats

Amb la comanda `subplot` es divideix la finestra de representació en tantes files i columnes com es senyali. Juntament amb la comanda `imshow` (per representació de la matriu com a imatge) i la comanda `title` (per mostrar un text damunt cada representació) s'obté una visió general dels resultats obtinguts així com les imatges generades per cada passa de l'algoritme.

A l'algoritme de processat principal d'aquest estudi es representen fins a 15 imatges diferents a la vegada per poder veure el procés i els resultats.

Els resultats es visualitzen automàticament a una finestra de representació gràfica de MATLAB. De la forma en que està definit l'algoritme, els resultats es visualitzen petits ja que es vol mostrar el funcionament d'una forma senzilla i per passes.

```
56 -    subplot(3,5,1);
57 -    imshow(f);
58 -    title('Original');
59 -    subplot(3,5,2);
60 -    imshow(g);
61      %imshow(hGray);
62 -    title('Filtre mediana 9x9');
63 -    subplot(3,5,3);
64 -    imshow(h);
65 -    title('Sharpen r = 10, a = 5');
66 -    subplot(3,5,4);
67 -    imshow(hBin);
68 -    title('Binarització');
69 -    subplot(3,5,5);
70 -    imshow(hBinFill);
71 -    title('imfill amb imatge invertida');
72 -    subplot(3,5,6);
73 -    imshow(iBin);
74 -    title('imclose (disk r=3) amb imatge invertida');
75 -    subplot(3,5,7);
76 -    imshow(hBinMed);
77 -    title('Filtre mediana 5x5 (no imfill)');
78 -    subplot(3,5,8);
79 -    imshow(hBinFill);
80 -    title('Filtre mediana 5x5 (amb imfill)');
81 -    subplot(3,5,9);
82 -    imshow(iBinMed);
83 -    title('Filtre mediana 5x5 (amb imclose)');
```

Figura 3.19: Part de representació visual de les imatges obtingudes

A l'apèndix J es pot veure la imatge J.1 amb les diferents representacions, damunt la qual s'expliquen els resultats.

3.3. Interpretació dels resultats

Com es pot observar a la imatge 3.20, els resultats de la binarització juntament amb la funció `imfill` provoquen una pèrdua d'informació que es determina important per a la resolució. Per tant, s'ha escollit la opció de `imclose` per aquest cas.

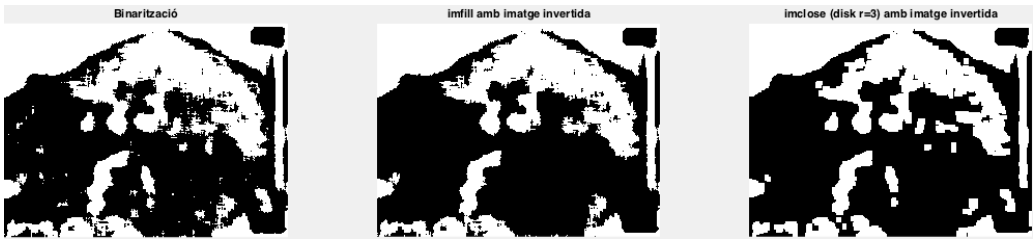

Figura 3.20: Imatges després de la binarització. Normal (esquerra), `imfill` (centre) i `imclose` (dreta)

Tant en el cas d'abans com en el de després d'aplicar el segon filtre de medianes, es pot veure que, encara que queden alguns punts que es podrien eliminar, la funció `imclose` funciona millor per aconseguir els resultats desitjats. A més la figura de l'aigua queda més natural en el contorn.

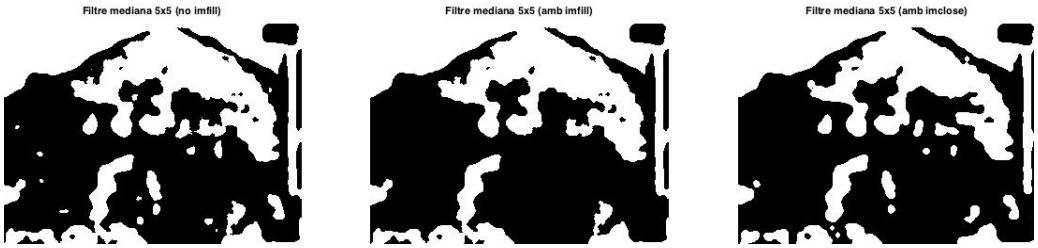

Figura 3.21: Imatges després de passar el segon filtre de medianes. Normal (esquerra), `imfill` (centre) i `imclose` (dreta)

A la figura 3.22 es pot veure el resultat de passar la funció `edge` a la imatge binaritzada tractada o no amb la funció `imclose`.

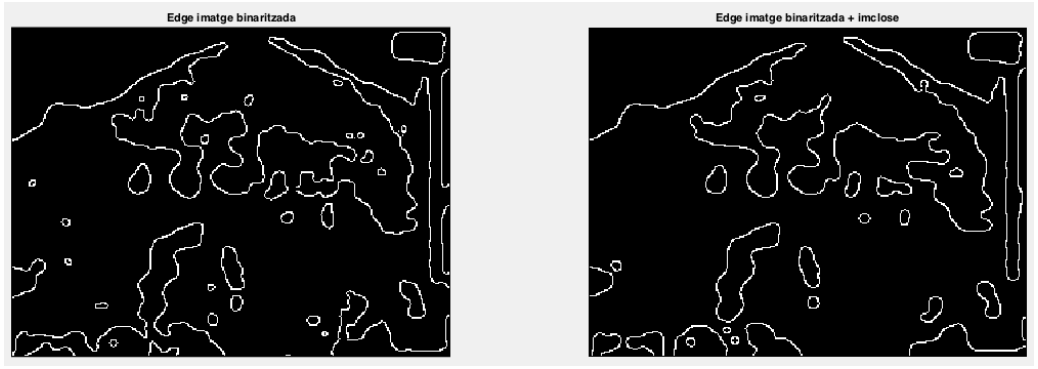

Figura 3.22: Imatges tractades amb la funció `edge`. Normal + filtre medianes (esquerra) i `imclose` + filtre medianes (dreta)

Comparant les imatges 3.23 y 3.24 s'observa que, tant a la imatge binaritzada com a la imatge del contorn (amb funció `edge`), s'aconsegueix una imatge més neta i definida si la imatge està tractada previament amb la funció `imclose`.

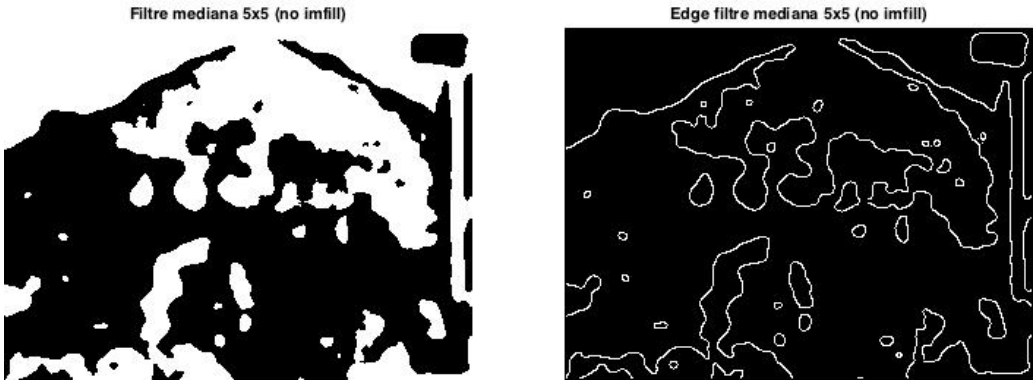

Figura 3.23: Imatges sense `imclose`

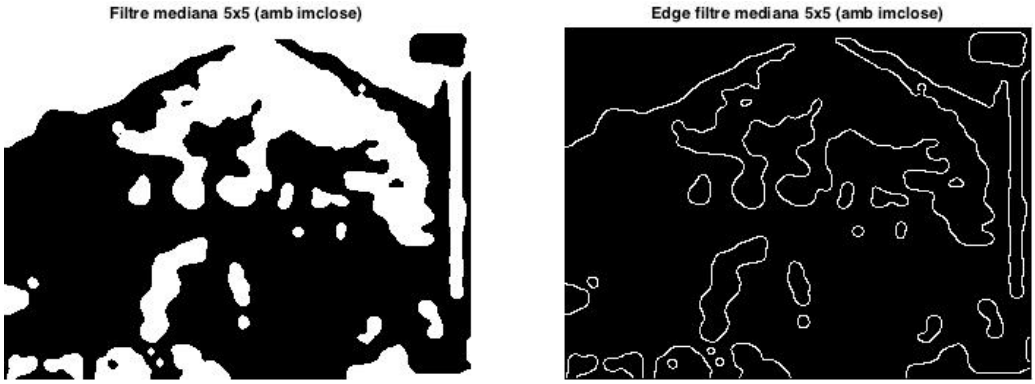

Figura 3.24: Imatges amb `imclose`

Finalment s'observa a la figura 3.25 la passa final de l'algoritme proposat. El resultat de la funció `edge` damunt la imatge binaritzada, juntament amb `imclose` i un segon filtre de medianes, es remarca a la imatge principal en color verd.

Es veu perfectament el recorregut de l'aigua i s'ajusta a les expectatives.

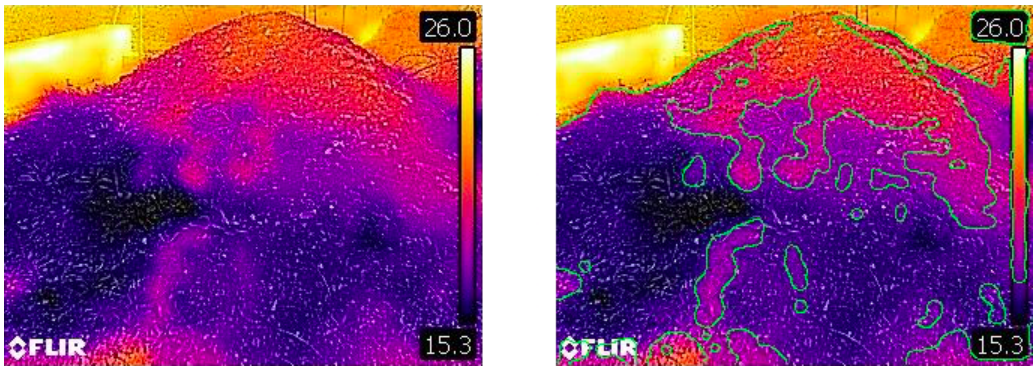

Figura 3.25: Resultat final remarcat en verd sobre la imatge inicial

CAPÍTOL 4

INTERFÍCIE GRÀFICA D'USUARI

4.1. *Graphic User Interface* (GUI)

MATLAB disposa d'eines que permeten crear una interfície d'usuari (GUI) personalitzada per a cada circumstància. L'objectiu del GUI implementat és aconseguir una interfície gràfica senzilla i versàtil per tal de posar l'algoritme a l'abast de equips d'investigació no massa versats amb la programació de MATLAB.

Donades les diverses variables de clima, temperatura i terreny que es poden donar en aquests casos, es fa imprescindible que els paràmetres dels filtres que conformen l'algoritme proposat siguin altament adaptables. Això, juntament amb una visualització de la imatge processada, servirà d'ajuda a un usuari no experimentat en tractament d'imatges per fer proves i aconseguir els millors resultats possibles.

També permetrà en el futur ampliar aquest estudi per aprendre el conjunt de paràmetres amb millors resultats i intentar l'automatització de tot el procés.

4.2. Algoritme GUI

Aquest algoritme és una modificació de l'algoritme de processat proposat al capítol 3 per tal d'adequar-lo a les especificacions de l'usuari. En aquest cas, es tracta d'una funció a la qual s'introdueix per paràmetre els valors de les variables emprades per l'algoritme, així com els activadors de les diferents parts. Es pot veure complet a l'apèndix H.

Es divideix en parts que equivalen a totes les funcions aplicables per l'usuari que estan controlades per variables *boolean* introduïdes per paràmetre.

L'inici de l'algoritme comprova si alguna operació principal és sol·licitada per l'usuari. En cas contrari, només retorna el valor d'entrada.

Les distintes parts de l'algoritme empren els valors introduïts a la funció per paràmetre. En cas de que l'usuari no determini el valors que vol emprar, es fan servir uns valors per defecte que han mostrat bons resultats a les imatges de prova.

algoritmeGUI.m | GUI_Imatges.m

```
1  function g = algoritmeGUI(f, temp, med1, shar, bin, fill, clo, med2, edg, rem, m1, r, a, t, m2)
2
3  %     figure;
4  %     subplot(3,3,1);
5  %     imshow(f);
6
7      if ~(med1||shar||bin||med2)
8          g = f;
9      else
10         if med1
11             h = RGBmedfilt2(f, m1);
12
13 %             subplot(3,3,2);
14 %             imshow(h);
15         end
16
17         if shar
18             h = imsharpen(h, 'radius', r, 'amount', a);
19
20 %             subplot(3,3,3);
21 %             imshow(h);
22         end
```

Figura 4.1: Declaració de la funció de l'algoritme per el GUI i funcionament dels condicionants

Com es veu a la figura 4.1, els paràmetres d'entrada de la funció es conformen de 9 variables *booleanes*, 5 variables numèriques i la imatge d'entrada a l'algoritme. La imatge d'entrada ve definida per la imatge cercada per l'usuari mitjançant un *browser* implementat al GUI.

Al final, la funció retorna la imatge processada.

L'algoritme inclou línies comentades, que corresponen a representacions de cada part de l'algoritme, per si es volgués veure el funcionament passa a passa.

En aquesta funció no s'ha implementat cap tipus de tractament d'errors i d'adaptació als paràmetres d'entrada ja que és una funció que no està pensada per ser cridada des de cap lloc que no sigui la interfície d'usuari. Per tant, com al GUI s'han definit paràmetres d'entrada fixos, la funció no rebrà cap tipus de dada que no sigui capaç de processar.

4.2.1. Aparença inicial

S'ha decidit implementar un únic algoritme que es divideix en parts que s'executen depenent de les decisions de l'usuari. La vista principal del GUI és d'aparença senzilla amb opcions de processat a l'esquerra i una pantalla de visualització a la dreta.

La pantalla inicial disposa d'un cercador d'arxius, juntament amb les parts principals de l'algoritme proposat per aquest estudi. No es mostren valors de parametrització ni els requadres per modificar-los, així com altres parts de l'algoritme que depenen directament de si es realitzen altres parts. Els botons de processat, *reset* i desar de les imatges apareixen bloquejats fins que es tria una imatge.

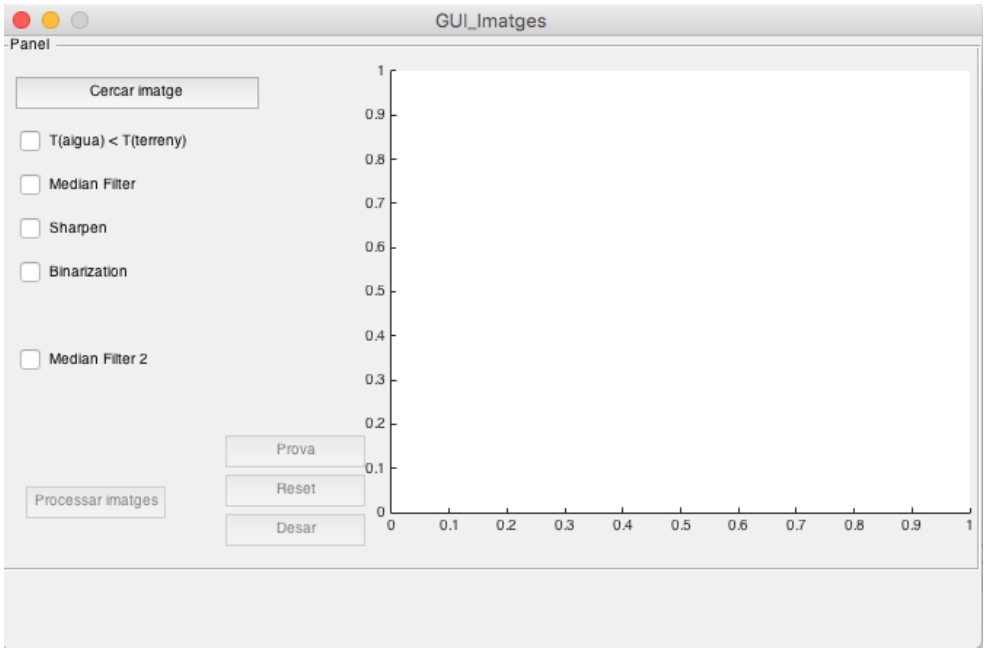

Figura 4.2: Aparença inicial del GUI

4.2.2. Funcionament

La primera passa serà seleccionar la imatge que es vol processar. Quan la imatge és seleccionada, es desbloquegen les opcions de processat i desat i es mostra la imatge a la pantalla de visualització.

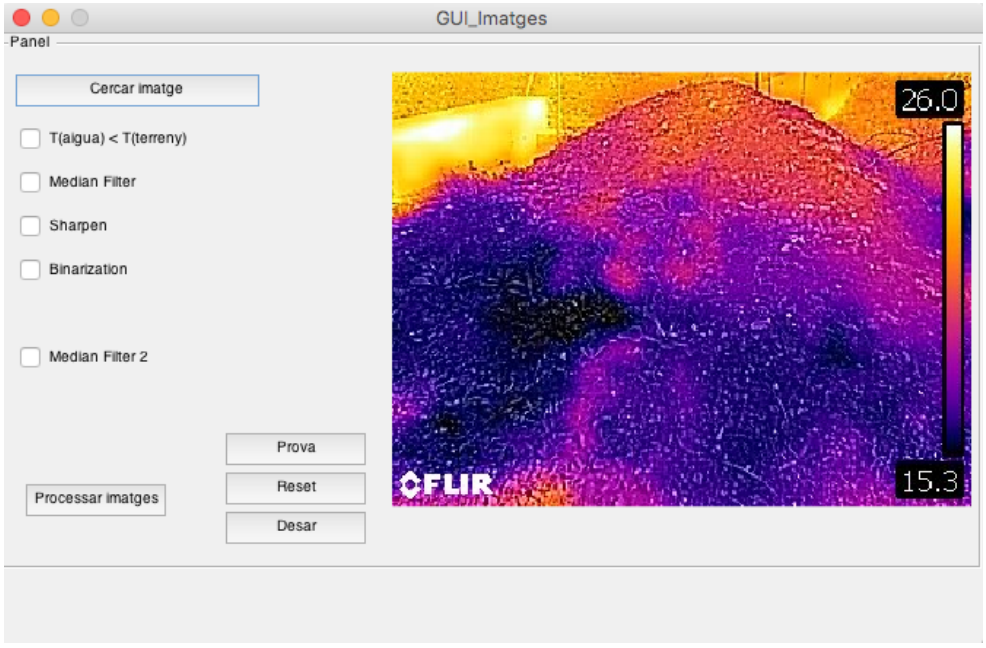

Figura 4.3: Mostra de la imatge després d'emprar el cercador. Resta d'opcions actives

En aquest punt, punt inicial del tractament de la imatge, les opcions de l'esquerra es poden activar i desactivar per personalitzar l'algoritme de processat. Es recomana sempre fer un filtre de medianes al principi amb una màscara gran (per exemple, 9x9) per tal de llevar o, al menys, disminuir el renou de la imatge.

La primera opció (T(aigua) < T(terreny)) fa referència a la temperatura de l'aigua en comparació a la temperatura del terreny. En cas de que l'aigua estigui mes freda que el terreny, s'ha de marcar aquesta opció per tal de invertir la imatge després de la binarització, i així aconseguir el mateix resultat d'aquest estudi: que l'aigua quedi representada en blanc.

L'opció de binarització és imprescindible si es volen desbloquejar la resta d'opcions de l'algoritme, les quals depenen d'una imatge binària. La opció *sharpen* és opcional, però recomanable, i el segon filtre s'emprarà per llevar el renou generat per el processament de la imatge i suavitzar el contorn de la forma resultant.

En marcar les opcions de l'esquerra s'activen les variables *booleanes* corresponents i desbloquegen les opcions de canvi dels valors dels paràmetres de cada funció. En cas de no introduir un valor als requadres o deixar-los en blanc, s'empren els valors assignats per defecte.

Les funcions *Fill* i *Close* son exclusives, es a dir, només una de les dues pot ser activada. La activació d'una desactivarà l'altre.

L'opció 'Remarcar en original' serveix per marcar els resultats de la imatge binaritzada o el contorn de la funció edge a la imatge original per comprovar si s'ajusta al que es cerca.

El botó 'Prova' realitza el processat de la imatge amb les característiques senyalades i mostra la imatge resultant. Es poden variar els valors dels paràmetres així com afegir o llevar opcions per tal d'adequar les característiques tantes vegades com es vulgui. El botó prova tornarà a fer el processat amb les noves assignacions i mostrarà el resultat.

El botó 'Reset' retorna els valors per defecte als paràmetres, desactiva totes les opcions i torna a mostrar la imatge inicial. S'ha implementat de forma que els requadres dels valors del paràmetre es quedin visibles i mostrin els valors per defecte per tal de saber quins valors s'han emprat. Així, si s'ha obtingut un resultat satisfactori, però no perfecte, (massa renou, pèrdua de zones importants,...), es sap mes o menys a partir de quins valors començar les proves. A la imatge 4.5 es pot observar com millora el resultat després de modificar els paràmetres.

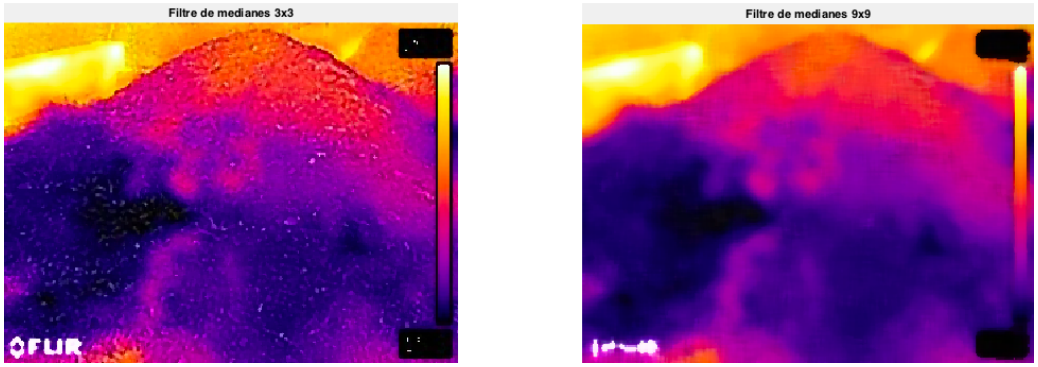

Figura 4.5: Imatge abans i després d'aplicar als paràmetres valors més adients

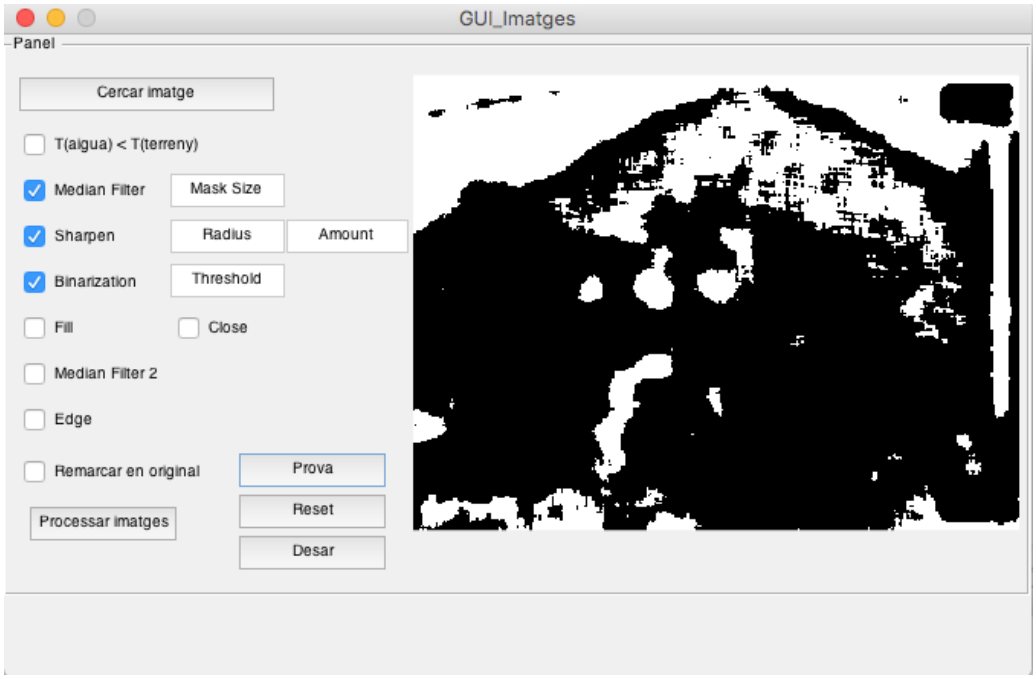

Figura 4.4: Funcionament de 'Prova' amb 3 opcions i valors per defecte

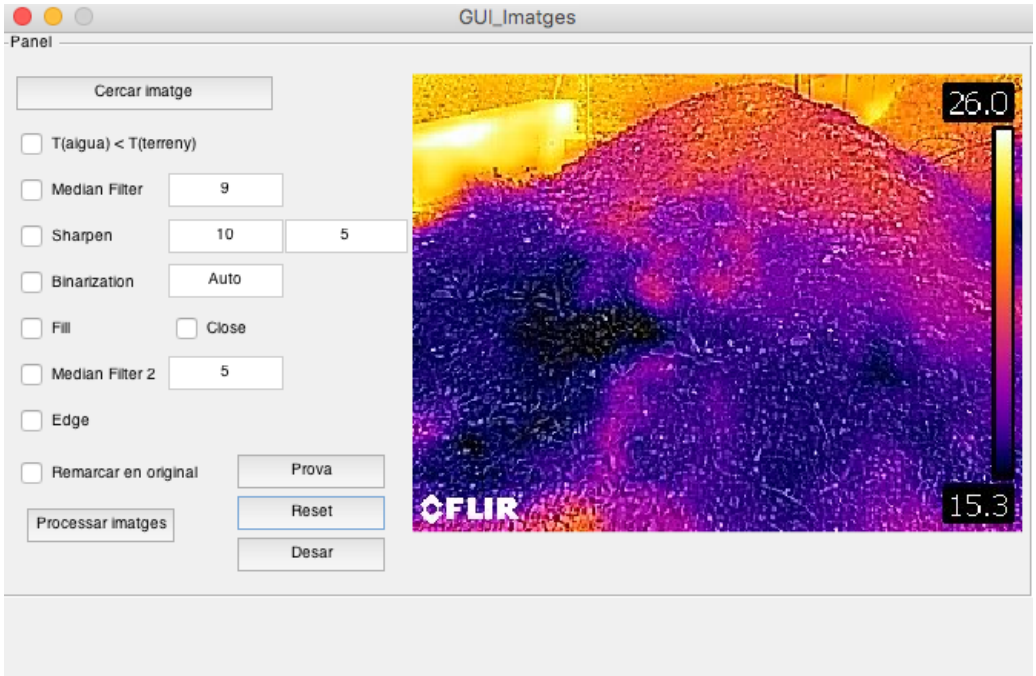

Figura 4.6: Mostra la interfície després de 'Reset'. Valors per defecte visibles

El botó 'Desar' permet guardar la imatge visualitzada al directori amb el nom que vulgui l'usuari. Pot ser una opció útil quan es vol comprovar resultats amb altres opcions o es volen guardar les passes del processat.

Finalment, el botó 'Processar imatges' permet a l'usuari seleccionar vàries imatges que seran processades amb les opcions senyalades i guardar-les al directori que l'usuari senyali amb un nom per defecte ('Processed_' més el nom d'arxiu de la imatge).

Encara que la imatge es representi a la interfície de mida inferior a la original, la imatge guardada és de la mateixa mida i dimensions (mateix nombre píxels) que la original.

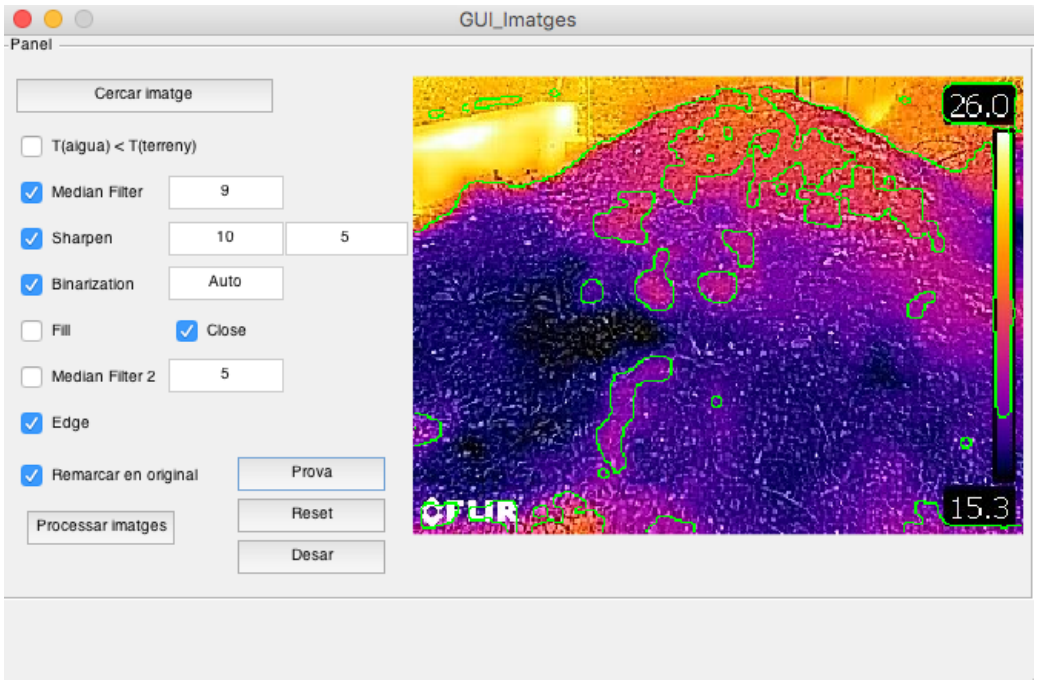

Figura 4.7: Possible solució amb algoritme complet i paràmetres personalitzats

4.2.3. Implementació del GUI

El GUI porta un sistema de funcions pròpies d'inicialització i funcionament. Les funcions principals del GUI venen definides per iniciar la interfície gràfica. S'han afegit línies de codi que corresponen a la inicialització de les variables i els seus valors per defecte. Es pot veure a la taula 4.1 les variables utilitzades i la seva funció.

Cada objecte afegit (*pushbutton*, *checkbox*, etc.) disposa d'una funció *callback* pròpia. Aquesta funció ha de ser modificada per tal d'implementar la seva funcionalitat.

Apart d'aquestes funcions *callback* s'han afegit funcions de modificació i lectura de variables globals, necessàries per ser emprades a distintes funcions.

4.2.4. Funcions de variables globals

Aquestes funcions s'implementen per els diferents valors que es modifiquen a unes funcions per ser emprades a altres. Per cada variable global es necessita una funció de lectura i una d'escriptura.

Per exemple, la variable *axes* serà la informació i direcció del sistema d'eixos cartesians, emprat per la representació de la imatge.

```
function setGlobalAxes(val)
global axes;
axes = val;
end

function x = getGlobalAxes
global axes;
x = axes;
end
```

Figura 4.8: Funcions d'escriptura i lectura de variable global *axes*

S'ha implementat una funció d'aquest tipus per cada variable numèrica dels paràmetres i per cada variable *boolean* que controlen l'execució per parts de l'algoritme. Es pot veure complet a l'apendix I.

4.2.5. Inicialització

A l'inici del GUI, és important definir les variables de control de l'algoritme així com donar-lis un valor per defecte. En cas de no activar cap funció, l'algoritme ha de conèixer que les variables de control son nul·les, així com en cas d'activar-les sense canviar el valors dels paràmetres, l'algoritme ha de tenir uns valors amb els que treballar.

Variable	**Valor inicial**		**Utlilitat**
		Variables boolean	
temp	false		El valor true inverteix la imatge binaritzada
med1	false		Activa o desactiva primer filtre de medianes
shar	false		Activa o desactiva `sharpen`
bin	false		Activa o desactiva binarització
fill	false		Activa o desactiva `imfill`
clo	false		Activa o desactiva `imclose`
med2	false		Activa o desactiva segon filtre de medianes
edg	false		Activa o desactiva `edge`
rem	false		Activa o desactiva `remarcar`
		Variables numèriques	
m1	9	*màscara 1*	Tamany màscara primer filtre de medianes
r	10	*radius*	Paràmetre *radius* de funció `sharpen`
a	5	*amount*	Paràmetre *ammount* de funció `sharpen`
t	auto	*threshold*	Valor *threshold* per binaritzasció
m2	5	*màscara 2*	Tamany màscara segon filtre de medianes

Taula 4.1: Variables de control de l'algoritme i els seus valors per defecte

La figura 4.9 mostra la funció que s'executa a la obertura del GUI (*opening function*). Es a dir, s'executa el moment abans de mostrar la interfície. Aquí és on s'han afegit les instruccions d'inicialització de les variables globals necessàries.

```
47    % --- Executes just before GUI_Imatges is made visible.
48    function GUI_Imatges_OpeningFcn(hObject, eventdata, handles, varargin)
49    % This function has no output args, see OutputFcn.
50    % hObject    handle to figure
51    % eventdata  reserved - to be defined in a future version of MATLAB
52    % handles    structure with handles and user data (see GUIDATA)
53    % varargin   command line arguments to GUI_Imatges (see VARARGIN)
54
55    % Choose default command line output for GUI_Imatges
56 -  handles.output = hObject;
57
58    % Update handles structure
59 -  guidata(hObject, handles);
60
61    % UIWAIT makes GUI_Imatges wait for user response (see UIRESUME)
62    % uiwait(handles.figure1);
63
64 -  setGlobalm1(9);
65 -  setGlobalr(10);
66 -  setGlobala(5);
67 -  setGlobalt(2);
68 -  setGlobalm2(5);
69 -  setGlobalmed1(false);
70 -  setGlobalshar(false);
71 -  setGlobalbin(false);
72 -  setGlobalfill(false);
73 -  setGlobalclo(false);
74 -  setGlobalmed2(false);
75 -  setGlobaledg(false);
76 -  setGlobalrem(false);
77
78 -  end
```

Figura 4.9: Funció d'*obertura* del GUI

4.2.6. Funcions *callback*

A partir de la funció d'obertura, el funcionament del GUI es realitza mitjançant les funcions anomenades *callback*. Son funcions que s'executen quan s'activen els diversos objectes de la interfície.

L'execució d'aquestes funcions pot ser donada per clicar un *checkbox* (caixes d'activació), canviar el valor d'un *text edit* (requadres de texte editables) o pitjar un dels botons de la interfície.

Aquestes funcions es creen en el moment d'afegir un objecte nou a la figura de la interfície. Les funcions apareixen en blanc i han de ser implementades segons la funcionalitat de cada objecte.

```
339     % --- Executes on button press in checkBin.
340   function checkBin_Callback(hObject, eventdata, handles)
341   % hObject    handle to checkBin (see GCBO)
342     % eventdata  reserved - to be defined in a future version of MATLAB
343   % handles    structure with handles and user data (see GUIDATA)
344
345     % Hint: get(hObject,'Value') returns toggle state of checkBin
346
347 -       setGlobalbin(get(hObject,'Value'));
348 -       h=findobj('tag','editThresh');
349 -       h1 = findobj('tag', 'checkFill');
350 -       h2 = findobj('tag', 'checkClose');
351 -       h3 = findobj('tag', 'checkEdge');
352 -       h4 = findobj('tag', 'checkRemarcar');
353
354 -       if get(hObject,'Value')
355 -           set(h, 'visible', 'on');
356 -           set(h1, 'visible', 'on');
357 -           set(h2, 'visible', 'on');
358 -           set(h3, 'visible', 'on');
359 -           set(h4, 'visible', 'on');
360
361 -       else
362 -           set(h, 'visible', 'off');
363 -           set(h1, 'visible', 'off', 'value', 0);
364 -           setGlobalfill(0);
365 -           set(h2, 'visible', 'off', 'value', 0);
366 -           setGlobalclo(0);
367 -           set(h3, 'visible', 'off', 'value', 0);
368 -           setGlobalrem(0);
369 -           set(h4, 'visible', 'off', 'value', 0);
370 -       end
371
372 -   end
```

Figura 4.10: Funció *callback* amb modificació de visibilitat i de valor de diferents objectes

Les funcions *callback* dels *checkbox* i els *text edit* normalment es basen en modificar un valor de les variables globals emprades per l'algoritme. Algunes també disposen de funcions de modificació de la visibilitat d'altres objectes o, com el cas de les funcions *fill* i *close*, implementacions d'exclusivitat.

Quan es lleva la visibilitat d'un objecte, també és necessari modificar els valors de les variables que impliquen els objectes afectats. Així mateix, l'estat dels mateixos objectes haurà de passar a desactivat o al seu valor per defecte. De la mateixa forma, la funció d'exclusivitat ha de desactivar l'altre funció i modificar les variables globals que afecten a la funció.

Les funcions *callback* dels botons han d'implementar distintes característiques segons les necessitats. Els botons 'Cercar imatge' i 'Processar imatges' permeten a l'usuari cercar arxius d'imatge per utilitzar a l'algoritme. Els botons 'Desar' i 'Processar imatges' demanen a l'usuari on es volen guardar les imatges processades.

Els botons 'Prova' i 'Processar imatge', a més, implementen la cridada de l'algoritme de processat.

Ordres	**Funció *callback***	**Acció**
		Opcions inicials
Cercar imatge	*pushbutton*	Obre una finestra per elegir una imatge
T(aigua)<T(terreny)	*checkbox*	S'ha de marcar quan l'aigua està més freda que el terreny. Activa la funció *complementar*
Median filter	*checkbox*	Activa/desactiva la variable med1 i activa Mask Size
Sharpen	*checkbox*	Activa/desactiva la variable shar i activa Radius i Amount
Binarization	*checkbox*	Activa/desactiva la variable bin i activa Threshold Permet l'accés a Fill i Close, a Edge i a Remarcar a l'original
Median filter 2	*checkbox*	Activa/desactiva la variable med1 i activa Mask Size
Prova	*pushbutton*	Executa l'algoritme de processat
Processar imatge	*pushbutton*	Processa un conjunt d'imatges totes a la vegada amb les opcions triades per l'usuari i les guarda a la carpeta elegida
Desar	*pushbutton*	Obre una finestre per guardar la imatge resultant
Reset	*pushbutton*	Reinicia tots els paràmetres, desactiva totes les opcions. Mostra la imatge inicial de nou
		Opcions que s'activen després de les opcions inicials
Mask Size	*text edit*	Defineix la mida de la màscara del primer filtre de medianes
Radius	*text edit*	Defineix el valor de Radius per Sharpen
Amount	*text edit*	Defineix el valor de Amount per Sharpen
Threshold	*text edit*	Defineix el valor de Radius per la binarització
Fill	*checkbox*	Activa/desactiva la variable fill i desactiva Close
Close	*checkbox*	Activa/desactiva la variable clo i desactiva Fill
Edge	*checkbox*	Activa/desactiva la variable edg
Mask Size 2	*text edit*	Defineix la mida de la màscara del segon filtre de medianes
Remarcar en original	*checkbox*	Activa/desactiva la variable rem

Taula 4.2: Els botons, les seves funcions *callback* i descripció del que fan

```
137    % --- Executes on button press in pushProces.
138    function pushProces_Callback(hObject, eventdata, handles)
139    % hObject    handle to pushProces (see GCBO)
140    % eventdata  reserved - to be defined in a future version of MATLAB
141    % handles    structure with handles and user data (see GUIDATA)
142
143 -      [file, lDir] = uigetfile(...
144            {'*.jpg; *.JPG; *.jpeg; *.JPEG; *.img; *.IMG; *.tif; *.TIF;'; ...
145            '*.tiff, *.TIFF','Supported Files (*.jpg,*.img,*.tiff,)'; ...
146            '*.jpg','jpg Files (*.jpg)';...
147            '*.JPG','JPG Files (*.JPG)';...
148            '*.jpeg','jpeg Files (*.jpeg)';...
149            '*.JPEG','JPEG Files (*.JPEG)';...
150            '*.img','img Files (*.img)';...
151            '*.IMG','IMG Files (*.IMG)';...
152            '*.tif','tif Files (*.tif)';...
153            '*.TIF','TIF Files (*.TIF)';...
154            '*.tiff','tiff Files (*.tiff)';...
155            '*.TIFF','TIFF Files (*.TIFF)'},...
156            'MultiSelect', 'on');
157 -      if ~(isnumeric(file) || isnumeric(lDir))
158 -          sDir = uigetdir('Desar imatges a...');
```

Figura 4.11: Part de interacció amb l'usuari de la funció *callback* de 'Processar imatges'

```
175 -               oldDir = cd(lDir);
176 -
177 -               i = size(file);
178 -
179 -               for j = 1:1:i(2)
180 -                   d = file{j};
181 -                   f = imread(d);
182 -
183 -                   g = algoritmeGUI(f, med1, shar, bin, fill, clo, med2, edg, rem, m1, r, a, t, m2);
184 -
185 -                   name = ['Processed_' d];
186 -
187 -                   cd(sDir);
188 -                   imwrite(g, name);
189 -                   cd(oldDir);
190 -               end
191 -           end
192 -       end
```

Figura 4.12: Part de lectura, execució de l'algoritme i escriptura de la funció *callback* anterior

CAPÍTOL 5

CONCLUSIONS

La popularització dels drons i la arribada d'aquests aparells a la investigació ha obert una possibilitat immensa a la fotografia aèria. És cada vegada més freqüent que els centres d'investigació i les universitats tinguin a la seva disposició aparells d'aquest tipus que permeten treure fotografies de forma ràpida i variada. Actualment és possible sortir a treure fotos d'una zona immediatament després d'un esdeveniment com per exemple, fortes pluges, un incendi o qualsevol fet que calgui investigar. Totes aquestes fotografies no seran d'utilitat si no se és capaç d'analitzar-les de forma àgil i amb una rapidesa equivalent a la facilitat que es té de recollida. En aquest treball s'ha desenvolupat una proposta de tractament d'imatges versàtil i eficaç. S'ha desenvolupat també una interfície gràfica que fa possible emprar les diferents alternatives sense necessitat de coneixements de programació i MATLAB mentre no es vulgui modificar.

5.1. Conclusions dels resultats

Amb l'algoritme emprat, es fa possible l'anàlisi de fotografies per l'observació dels fluxos d'aigua així com les zones de drenatge de l'aigua. Aquests resultats es poden observar fàcilment a la figura 5.1. Es pot veure el flux de l'aigua remarcat en verd damunt la imatge, així com es pot determinar les zones de drenatge on l'aigua passa a un flux subterrani i on torna a sortir a la superfície.

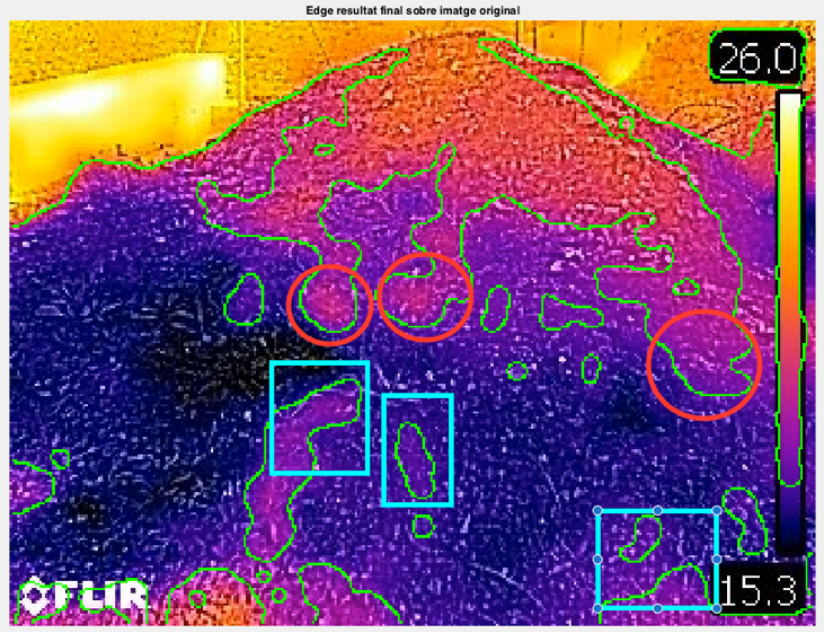

Figura 5.1: Zones de drenatge i sortida de l'aigua remarcades (cercles vermells i rectangles blaus respectivament).

El resultat obtingut ens permet visualitzar clarament els objectius d'aquest treball. Per tant, es pot dir que és un mètode totalment vàlid per l'observació de fluxos d'aigua i zones de drenatge.

5.2. Utilitats de l'estudi

Aquests resultats poden ser de gran ajuda per la hidrologia, tant per l'estudi dels fluxos pluvials com per la determinació de zones de drenatge i possibles fluxos d'aigües subterrànies.

També és útil a l'àmbit de la agricultura. Aquest estudi pot ser essencial per determinar les zones on l'aigua s'acumula de forma natural i les zones on la terra no rebrà aigua i haurà de ser transportada artificialment.

Els drons ja son emprats per el sector agrícola, per tant afegir aquesta tècnica de mapeig pot no significar una gran inversió i resultar en un gran avantatge per el desenvolupament de les tècniques agrícoles actuals.

APÈNDIX A

ESPECIFICACIONS CÀMERA TÈRMICA

Número de modelo	**FLIR C2**
	Datos ópticos y de imagen
Sensor IR	80 x 60 (4800 píxeles de medición)
Sesibilidad térmica	$< 0,10°C$
Campo de visión	41° x 31°
Distancia focal mínima	Termografía: 0,15 m (0,49 pies) MSX®: 1,0 m (3,3 pies)
Frecuencia de imagen	9 Hz
Rango espectral	7,5 - 14 μm
Pantalla de 3" (color)	320 x 240 píxeles
Orientación automática	Sí
Pantalla táctil	Sí
	Modos de presentación de imagen
Termografía	Sí
Imagen visual	Sí
MSX®	Sí
Galería	Sí
	Medición
Intervalo de temperaturas del objeto	De -10° C a +150º C (de 14 a 302° F)
Precisión	±2° C (±3,6° F) o 2%, la que sea superior, a 25° C (77° F) nominal
	Análisis de medición
Fotómetro puntual	Encendido/apagado
Corrección de emisividad	Sí; mate/semi/brillante + configurable
Corrección de medidas	Emisividad de temperatura aparente reflejada
	Configuración
Paletas de colores	Hierro, Arco iris, Arco iris HC, Gris
Soporte de almacenamiento	Memoria interna con capacidad para almacenar al menos 500 conjuntos de imágenes
Formato de archivo de imagen	JPEG estándar, datos de medición de 14 bits incluidos
	Transmisión de vídeo
Transmisión de vídeo de infrarrojos no radiométrico	Sí
Transmisión de vídeo visual	Sí
	Cámara digital
Cámara digital	640 x 480 píxeles
Cámara digital, enfoque	Enfoque fijo
	Información adicional
©	USB Micro-B: Transferencia de datos desde y hacia sistemas Windows, iOS y Android
Batería	Batería recargable de 3,7 V de polímero de iones de litio
Duración de la batería	2 horas
Sistema de carga	Se carga en la cámara
Tiempo de carga	1,5 horas
Funcionamiento con alimentación externa	Adaptador de CA, entrada de 90 - 260 V CA Salida de 5 V a la cámara
Gestión energética	Apagado automático
Intervalo de temperatura de funcionamiento	De -10° C a +50°C (de 14 a 122°F)
Intervalo de temperatura de almacenamiento	-40° C a +70° C (?40 a 158° F)
Peso	0,13 kg (0,29 lb)
Tamaño (L x An x Al)	125 x 80 x 24 mm (4,9 x 3,1 x 0,94 pulgadas)

Taula A.1: Especificacions FLIR C2. (Font: http://www.flir.es/instruments/c2/)

APÈNDIX B

MATLAB – NOCIONS BÀSIQUES

En aquest annex s'expliquen els operadors i les comandes bàsiques de MATLAB per poder comprendre el seu funcionament a les capes més baixes de processat de dades.

B.1. Introducció a MATLAB

El llenguatge M de MATLAB és un llenguatge d'alt nivell orientat a objectes similar a *Python*.

En el cas de MATLAB no és necessari declarar les variables ni els arguments de les funcions de cap tipus abans de ser emprats. Una variable o argument és declara com el tipus del valor assignat en el moment. Es pot veure un exemple a la figura B.1.

El comandament `whos` és emprat per saber tota la informació referent a una funció o variable. A la figura B.1 veiem el seu funcionament. Es dona valor a 4 variables diferents (a, b, c i d) amb tipus de dades diferents. Es veu els diferents tipus de declaració de variables per *arrays* de caràcters, valors numèrics i matrius.

Igualment és molt senzilla la conversió dels tipus de variables, podem transformar un valor en coma flotant (`double`) en un sencer positiu de 8 / 16 / 32 / 64 bits (*uint8* / *uint16* / *uint32* / *uint64*) amb un simple comandament com podem veure a la figura B.2.

Com veiem a la figura B.2, continuant des de la figura B.1, la classe de `d` i `c` canvien fàcilment indicant el tipus de valor en que es volen convertir. Es pot observar que a l'hora de passar un *array* de caràcters a *array* de sencers, els valors dels nombres sencers agafen el valor numèric decimal del codi ASCII dels caràcters.

B.2. Operadors de MATLAB

Operadors aritmètics: Aquests operadors s'empren exclusivament per la realització d'operacions aritmètiques com la suma, resta, multiplicació, etc. Taula B.1.

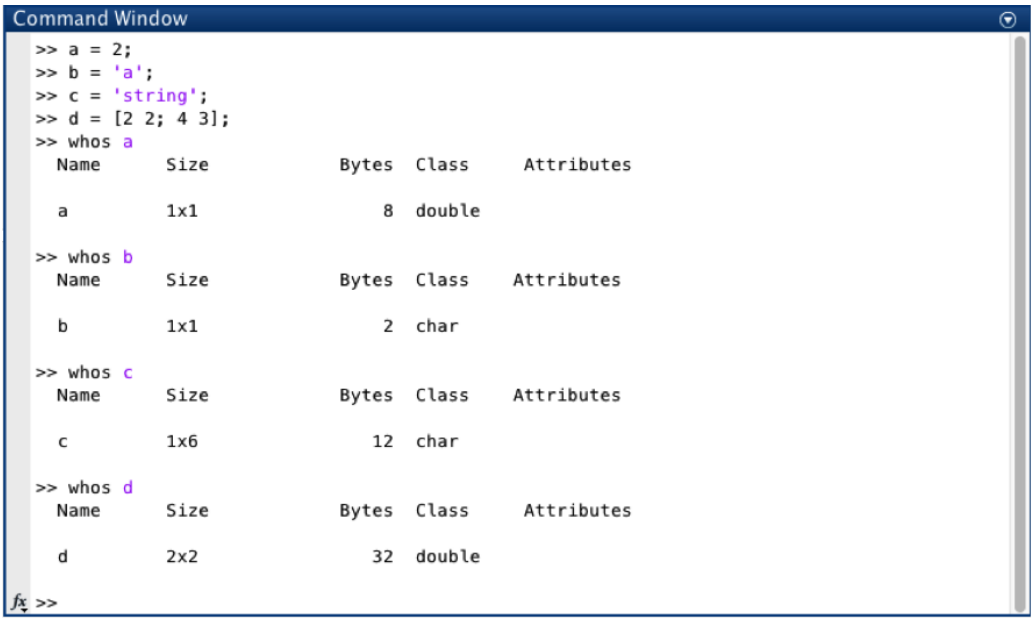

Figura B.1: Declaració i definició de variables a MATLAB

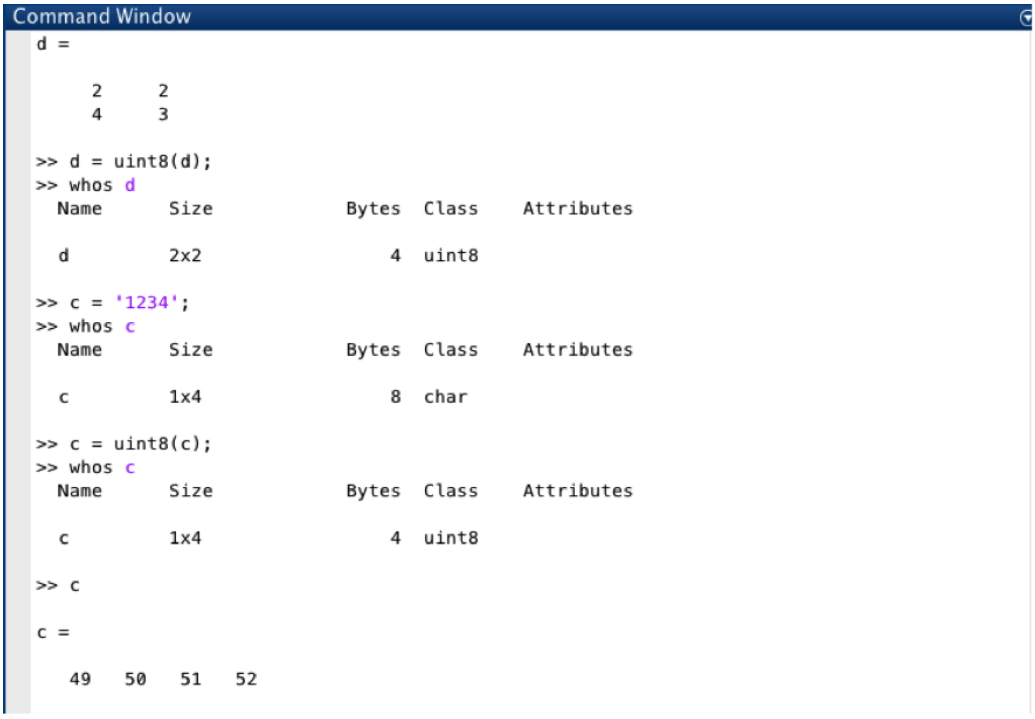

Figura B.2: Canvis del tipus de variable a MATLAB

Aritmètics		
Operador	**Símbol**	**Exemple**
Suma	+	3 + 2
Resta	−	3 − 2
Multiplicació	*	3 * 2
Divisió	/	3/2
Divisió inversa	\	2 \ 3 = 3/2
Potència	^	3^2

Taula B.1: Operadors aritmètics

Operadors lògics: Aquests operadors s'empren per el desenvolupament d'expressions lògiques emprades per operar amb condicionants (`if/else`) o bucles (`while`). Taula B.1.

Lògics		
Operador	**Símbol**	**Descripció**
And	&	Retorna vertader si ambdues proposicions son vertaderes
Or	|	Retorna vertader si una de les dues proposicions es vertadera
Not	~	Retorna vertader si la proposició següent es falsa

Taula B.2: Operadors lógics

Operadors relacionals: Aquests operadors s'empren per a comparació de valors matemàtics o lògics. Retornen valor "vertader" (1) si es compleix. Taula B.3.

Relacionals		
Operador	**Símbol**	**Exemple**
Major	>	3 > 2 → 1
Menor	<	3 < 2 → 0
Igual	==	3 == 2 → 0
Major o igual	>=	3 >= 2 → 1
Menor o igual	<=	2 <= 3 → 0
Distint	~=	3~= 2 → 1

Taula B.3: Operadors relacionals

B.3. Estructures condicionals

`if / elseif / else`

Estructura condicional equivalent a *si / si no, si / si no*. És emprada per dividir el codi en blocs condicionats a les expressions que segueixen el comandament `if` o `elseif`.

Per acabar el bloc s'empra el comandament `end`.

El bloc en qüestió només s'executarà si l'expressió es certa. Veiem un exemple a la figura B.3.

```
>> a = 10;
b = 5;
if a > b
    c = a - b;
elseif a < b;
    c = a + b;
    else
        c = a * b;
end
c

c =

     5
```

```
>> a = 5;
b = 10;
if a > b
    c = a - b;
elseif a < b;
    c = a + b;
    else
        c = a * b;
end
c

c =

    15
```

```
>> a = 10;
b = 10;
if a > b
    c = a - b;
elseif a < b;
    c = a + b;
    else
        c = a * b;
end
c

c =

   100
```

Figura B.3: Estructura `if / elseif / else`.

`switch / case`

Estructura condicional de casos relacionats amb una variable. El valor que segueix el comandament `switch` pot tindre diferents valors, que permetran l'execució del bloc `case` corresponent. Si el valor d'entrada no es correspon amb cap `case` s'executarà el bloc `otherwise`. També es necessari acabar l'estructura amb el comandament `end`. Veiem un exemple a la figura B.4.

```
>> c = 'a';

switch c
    case 'a'
        disp('vocal "a"')
    case 'b'
        disp('consonant "b"')
    otherwise
        disp('no és "a" ni "b"')
end
vocal "a"
```

```
>> c = 'b';

switch c
    case 'a'
        disp('vocal "a"')
    case 'b'
        disp('consonant "b"')
    otherwise
        disp('no és "a" ni "b"')
end
consonant "b"
```

```
>> c = 'c';

switch c
    case 'a'
        disp('vocal "a"')
    case 'b'
        disp('consonant "b"')
    otherwise
        disp('no és "a" ni "b"')
end
no és "a" ni "b"
```

Figura B.4: Estructura `switch / case`.

B.4. Estructures de bucle

Bucle `for`

L'estructura del bucle `for` es basa en un comptador que recorre els valors fins a arribar a un màxim definit. A cada iteració del bucle s'incrementa el comptador en l'interval definit.

Les condicions de la sentència `for` es defineixen de la següent forma: `for i = valorInicial:interval:valorFinal`. Després d'escriure el bloc d'iteració, s'acaba amb `end`. Podem veure un exemple a la figura B.5.

```
>> for i = 1:1:5
    disp(i);
end
     1

     2

     3

     4

     5
```

Figura B.5: Canvis del tipus de variable a MATLAB.

Bucle `while`

L'estructura d'aquest bucle es basa en fer iteracions mentre la condició que segueix a la sentència `while` sigui certa. També s'ha d'acabar el bloc d'iteració amb el comandament `end`.

Veiem un exemple equivalent al bucle `for` anterior a la figura B.6.

```
>> i = 1;
while i <= 5
    disp(i);
    i = i + 1;
end
     1

     2

     3

     4

     5
```

Figura B.6: Canvis del tipus de variable a MATLAB.

APÈNDIX C

MATLAB – IMAGE PROCESSING TOOLBOX

Ja s'ha parlat de l'existència d'aquesta *toolbox* anteriorment. En aquest annex s'explicaran i compararan les diferents funcions emprades per al processament digital d'imatges. L'objectiu es fer una breu introducció a les funcions bàsiques de lectura i escriptura d'imatges així com explicar els tipus de filtres que es podrien emprar per la situació d'estudi. Aquests filtres es compararan pràcticament mitjançant exemples damunt imatges tèrmiques i es triaran els millors processos per crear un algoritme de proves.

C.1. Lectura, escriptura i representació

Les funcions per aquest apartat son bastant intuïtives. Serveixen per llegir un arxiu d'imatge i guardar-lo en una variable de tipus *array*, mostrar aquesta imatge per pantalla i guardar la imatge definida per una variable en un directori amb el nom escollit. Veiem un exemple d'aquestes sentencies a les figures C.1 i C.2.

```
f = imread('prueba.jpg');

subplot(1,4,1);
imshow(f);
title('Imatge original');
subplot(1,4,2);
imshow(f(:,:,1));
title('Capa vermella');
subplot(1,4,3);
imshow(f(:,:,2));
title('Capa verda');
subplot(1,4,4);
imshow(f(:,:,3));
title('Capa blava');

oldDir = cd('/Users/Dani/Desktop');
imwrite(f, 'ProvaCompletada.jpg');
cd(oldDir);
```

Figura C.1: Codi lectura, escriptura i representació d'imatges

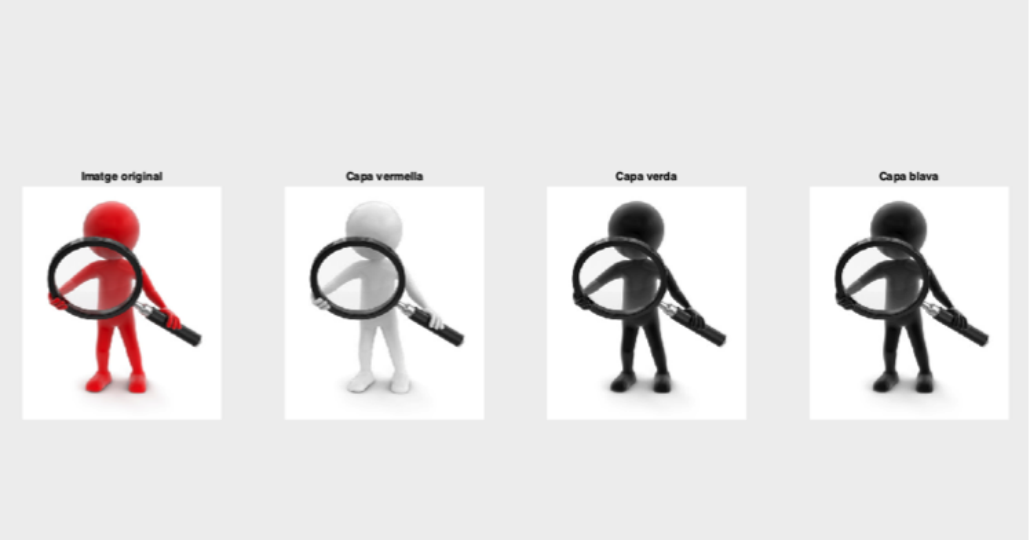

Figura C.2: Resultat lectura, escriptura i representació d'imatges. (Font imatge original: El blog del notario - https://pildoraslegales.com)

Per explicar el codi anterior començarem per explicar les diferents sentencies emprades:

`imread(NomImatge.ext)`

Llegeix una imatge des d'un fitxer i la guarda dins una variable en forma d'*array*. Es defineix l'extensió al mateix nom de la imatge. Si la imatge té format de capes (com .jpg) es guarda com *array* tridimensional de dades. Per guardar la imatge i no

només mostrar els seus valors és necessari igualar una variable I a la funció: `I = imread(NomImatge.ext)`.

`subplot(files,columnes,posició)`

Quan es parla de representar funcions, o imatges en aquest cas, `subplot` ens permet dividir la finestra de representació en diverses parts igualitàries per representar vàries imatges a l'hora.

Els tres arguments que s'introdueixen a la funció son el nombre de files/columnes que tindrà la divisió de la finestra (dos primers arguments) i la posició on volem representar o mostrar la figura actual, com a un nombre que va de 1 al producte de files i columnes. S'omplirà primer la primera fila d'esquerra a dreta fins acabar les files.

`imshow(I)`

Aquesta funció serveix per tractar una variable en forma d'*array* com una imatge i representar-la a la secció definida per `subplot`.

`title(Títol)`

Permet donar títol a la imatge o funció representada. El títol es col·loca a la part superior de la secció indicada per el darrer `subplot`.

`cd(Directori)`

Canvia el directori de treball de MATLAB a la direcció definida com `string` introduïda per paràmetre. En cas de igualar una variable a aquesta funció, la variable agafarà com a valor un `string` equivalent al directori anterior al canvi. Això servirà per retornar a l'antic directori per continuar la feina.

`imwrite(I,NomImatge.ext)`

Aquesta es la versió contraria de `imread()`. Guarda en el directori de treball la imatge I (variable de tipus *array* de dades) amb el nom introduït com `string`. És necessari completar el nom de la imatge amb l'extensió amb que es vol desar la imatge.

Una vegada vistes aquestes funcions és fàcil entendre la representació de la figura 4.2. Veiem que mitjançant `subplot` hem creat 4 seccions a la finestra de representació on hem representat la imatge original a la primera posició i les 3 capes que la conformen a la resta de seccions. Per accedir a les capes de la imatge, ho fem de la mateixa forma que es fa referència a les capes d'un *array* tridimensional.

Donat un *array* tridimensional f accedim a la primera capa de la següent forma: `f(:,:,1)`. El signe "`:`" expressa 'tots els valors'. Per tant, els primers dos paràmetres fent referència a files i columnes respectivament, d'aquesta forma es criden els valors de totes les files i totes les columnes de la primera dimensió de l'*array*. Com veiem cada títol introduït amb la funció `title` s'escriu damunt la secció creada per el darrer `subplot`. D'aquesta forma s'aconsegueix una visualització fàcil i ràpida per a comparar els resultats d'un estudi.

El signe "`;`" serveix per no mostrar els valors de les operacions a la finestra de comandaments.

C.2. Canvis de tipus d'imatge

Quan es parla del tipus d'una imatge es fa referència a la forma en que es defineixen els colors i la intensitat de les imatges. Els tipus d'imatge es divideixen en: imatges d'intensitat, imatges indexades i imatges de color real (o imatges RGB).

Imatges d'intensitat (*intensity images*):

Les imatges d'intensitat es defineixen amb valors dins un interval definit a cada cel·la o píxel. Aquests valors fan referència a la intensitat de color que representen. Encara que aquest tipus d'imatges només es guarden amb la matriu d'intensitats, necessiten d'un "mapa de colors" (o *colormap*) per a ser representades en tonalitats que no siguin grises, tractant-se igual que les imatges indexades (la intensitat passa a ser un índex del *colormap*).

Aquestes imatges poden ser representades amb valors `double` (0-1), `uint8` i `uint16`.

Es tracta d'imatges com les anomenades imatges en escala de grisos, és a dir, imatges on només es regula la intensitat del gris a representar en cada punt. Un exemple es mostra a la figura C.3. També són imatges d'intensitat les imatges binaries en blanc i negre, on cada punt té el valor sencer 0 o 1.

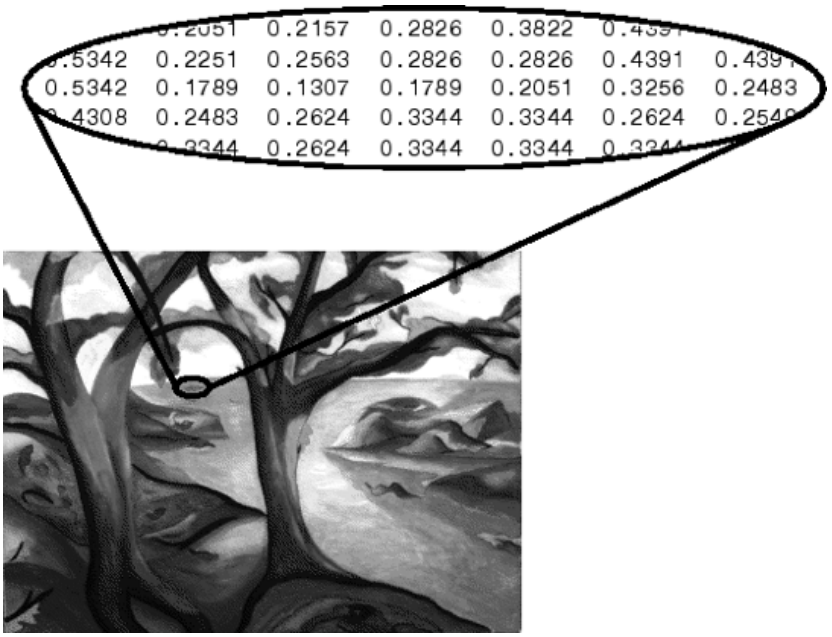

Figura C.3: Imatge de intensitats en escala de grisos. (Font: es.mathworks.com)

Imatges indexades (*indexed images*)

Les imatges indexades disposen d'una matriu d'índexs que fan referència a una matriu de "mapa de colors" definit. El *colormap* és una matriu de mida Mx3 (M és la quantitat de colors diferents a la imatge) on els grups de 3 files representen els tres

colors (vermell, verd i blau) per conformar cada color present a la imatge. Per tant una imatge indexada disposa de dues matrius: una que defineix els índexs del color al que fan referència i una que defineix els color possibles. Figura C.4.

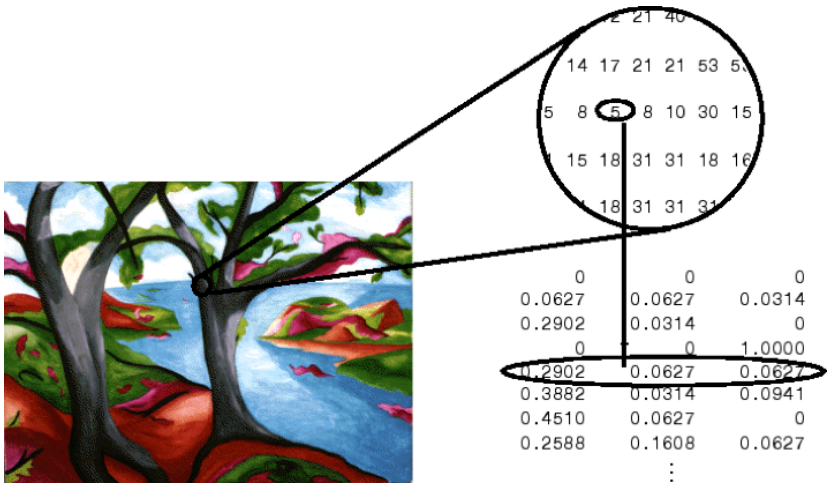

Figura C.4: Imatge indexada en color RGB. (Font: es.mathworks.com)

Imatges de color real (*true color images*) o RGB

Les imatges de color real son descrites, com ja s'ha explicat en capítols anteriors com una matriu tridimensional de tres capes de dimensió MxN. Cada capa representa un dels tres colors primaris per a la síntesi de color amb addició de llum (vermell, verd, blau o *Red*, *Green*, *Blue* en anglès). Figura C.5.

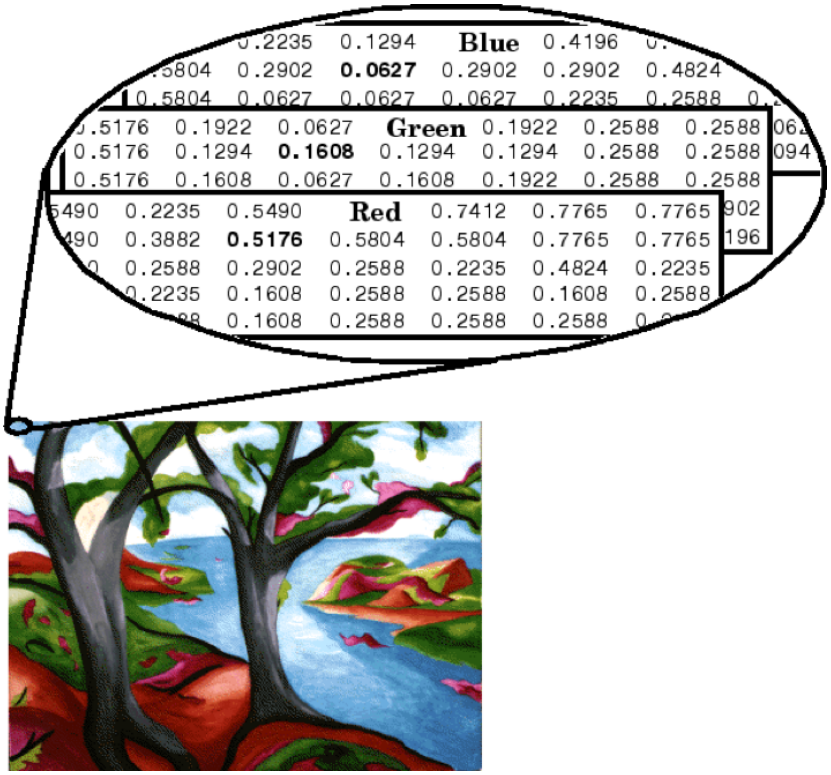

Figura C.5: Imatge indexada en color real RGB. (Font: es.mathworks.com)

Aquest treball es basa en el processament d'imatges tèrmiques a color. Per tant tractarem amb imatges de tipus RGB. El format en que la càmera *FLIR C2* guarda les imatges es *.jpg*.

Com a les imatges que es disposen per aquest treball no disposem d'una temperatura fixa (que es tradueix en un color fixe), ens interessa sobretot la diferència de intensitats a la imatge. Quanta més intensitat més ens acostem al blanc (que significaria la màxima temperatura a una imatge tèrmica).

Per tant, és molt útil canviar el tipus de la imatge d'entrada a un tipus de imatge d'intensitats. El tipus d'imatge que cerquem és una imatge en escala de grisos (*grayscale image*). Això s'aconsegueix amb la funció de MATLAB `rgb2gray()`.

```
g = rgb2gray(f)
```

Defineix `g` com una imatge en escala de grisos a partir de la intensitat dels valors de la imatge RGB `f`. Aconseguida la imatge d'intensitats, es sap que l'aigua està a menys temperatura que el terreny, però depenent de la quantitat d'aigua que passa per el terreny, es refredarà més o menys la terra per on circula el flux d'aigua. Com l'interès d'aquest estudi és trobar els recorreguts de l'aigua, i no un conjunt d'aigua immòbil, s'han d'observar diferents temperatures, segons la quantitat d'aigua que passi per els distints recorreguts.

Per tant, ens interessa observar a partit de quins valors de la imatge interpretem que es tracta d'un flux d'aigua. És convenient observar l'histograma de la matriu imatge

per veure quins valors son els més comuns i determinar a partir de quina temperatura o quins valors de intensitat volem definir l'aigua o els recorreguts.

`histogram(g,n)`

Mostra en intervals de 5 (en cas de valors *uint8* [0-255]) la quantitat de cel·les que tenen un valor dins cada interval. En cas d'especificar un valor de n, es faran n subdivisions en intervals de la quantitat de valors dividit entre n.

A partir de l'histograma podem definir quins valors interpretarem com aigua. Fet això, és interessant convertir la imatge de intensitats en una imatge binària (de valors lògics 0 i 1). Per això emprarem la funció `im2bw()`, que ens permet transformar tots els valors de una imatge de intensitats en valors lògics. L'únic que s'ha de tindre en compte és a partir de quins valors de la imatge s'agafarà valor lògic alt (1). Per tant, observant l'histograma podem definir el valor del `threshold`, un valor entre 0 i 1 que determina a partir de quin valor de intensitat entre 0 i 1 es passa a valor alt (1).

La funció `im2bw()` tracta la imatge de intensitats com a valors double entre 0 i 1 encara que la imatge tingui valors uint8 entre 0 i 255, així que no ens preocupa haver de transformar el tipus de dades de la imatge antes d'emprar aquesta funció, però sí el valor del `threshold` que definim a partir de l'histograma. Simplement es dividirà el valor escollit entre 255 per deixar-lo en escala 0-1.

`h = im2bw(g,T)`

Defineix `h` com una imatge binària de `g` a partir del valor `T`(`threshold`). Aquest valor `T` també es pot definir automàticament mitjançant la funció `graythresh(I)`. Aquesta funció calcula el `threshold` per a la conversió de la imatge a partir del mètode de Otsu.

El mètode de Otsu és un mètode que es basa en l'observació de l'histograma i tria el valor del `threshold` (k) que maximitza la variància entre classes σ_B^2, que es defineix a la expressió C.1. [13]

$$\sigma_B^2 = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2 \tag{C.1}$$

Eq. 4.1. Variància entre classes. Mètode de Otsu

Els termes de l'expressió venen definits a les següents expressions C.2, C.3, C.4, C.5 i C.6.

$$\omega_0 = \sum_{q=0}^{k-1} p_q(r_q) \tag{C.2}$$

Eq. 4.2. Mètode de Otsu - 1

$$\omega_1 = \sum_{q=k}^{L-1} p_q(r_q) \tag{C.3}$$

Eq. 4.3. Mètode de Otsu - 2

$$\mu_0 = \sum_{q=0}^{k-1} \frac{q \cdot p_q(r_q)}{\omega_0} \tag{C.4}$$

Eq. 4.4. Mètode de Otsu - 3

$$\mu_1 = \sum_{q=k}^{L-1} \frac{q \cdot p_q(r_q)}{\omega_1} \tag{C.5}$$

Eq. 4.5. Mètode de Otsu - 4

$$\mu_T = \sum_{q=0}^{L-1} q \cdot p_q(r_q) \tag{C.6}$$

Eq. 4.6. Mètode de Otsu - 5

on $p_q(r_q)$ és la funció de densitat de probabilitat discreta per l'histograma normalitzat. Es pot veure a l'expressió C.7.

$$p_q(r_q) = \frac{n_q}{n} \qquad q = 0, 1, 2, ..., L-1 \tag{C.7}$$

Eq. 4.7. Mètode de Otsu - 6

on n és el nombre total de píxels de la imatge, n_q el nombre de píxels que tenen un valor de intensitat r_q, i L és la quantitat total de possibles intensitats.

Aquest mètode, al nostre cas, no sempre serà útil. Fallarà als casos on els recorreguts d'aigua siguin petits. Per tant, haurem de fer proves variant manualment el `threshold`, però serà convenient mirar l'histograma per agafar una referència.

Observant l'histograma podrem veure els intervals amb valors de intensitat (en aquest cas temperatura) similars que tenen alt nombre de píxels d'aquest valor. Després, a partir d'aquí es faran proves manualment per determinar el `threshold` òptim per aquesta imatge. Veiem un exemple d'histograma a la figura C.6.

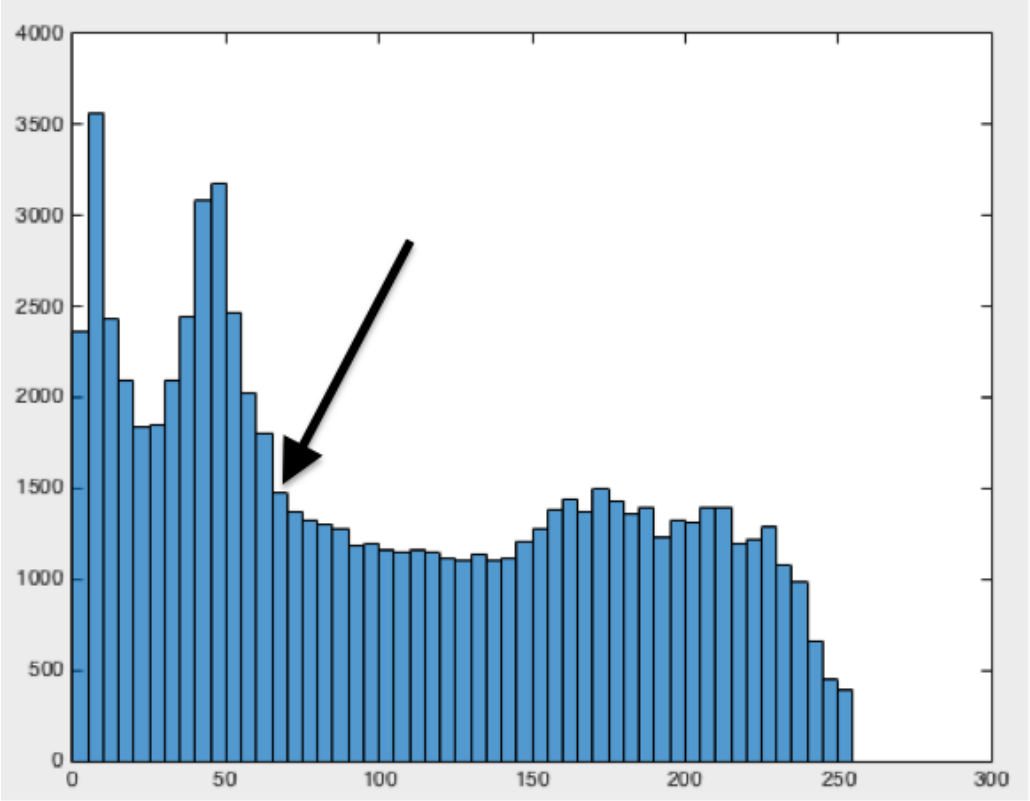

Figura C.6: Histograma imatge tèrmica

A partir d'on es senyala amb la fletxa els valors son bastant constants, per tant serà un bon valor per començar la prova del `threshold`. El valor senyalat és 65, però com necessitem un valor entre 0 i 1 el dividirem per el valor màxim de intensitat 255. Per tant, començam les proves a partir de un valor `threshold` de 0.255

C.3. Filtres a MATLAB

Ja s'ha explicat el funcionament dels filtres a seccions anteriors, per tant, a aquest apartat s'explicaran els diferents filtres de MATLAB emprats per aquest treball.

`ordfilt2`

Aquest filtre substitueix el valor del píxel en qüestió per el valor d'ordre n indicat entre els valors dels píxels de la màscara de treball que no siguin 0. La sintaxi de cridada d'aquest filtre és:

```
g = ordfilt2(f,order,domain,padopt)
```

on `g` és la sortida (imatge filtrada), `f` és la imatge inicial sense filtrar, `order` és la posició de menor a major del valor que s'agafarà per substituir el píxel d'estudi, `domain` és la màscara de treball en forma de matriu i `padopt` son les opcions de `padding` de les

vores (`zeros` per `padding` amb `zeros` i `symmetric` per `padding` amb simetria de mirall).

Cal dir que només es tindran en compte els valors de la matriu màscara que no siguin zero, per tant en aquest filtre es poden excloure valors d'una finestra manualment. Tot i així es sol emprar una matriu unitat quadrada per l'atribut domain de la funció (`ones(N)`: crea una matriu unitat NxN).

Alguns exemples són:

1. `g = ordfilt2(f,1,ones(3))`. Es tracta d'un filtre de mínims on s'agafa el valor més petit dels que cobreix la màscara. Finestra 3x3 i opcions de `padding` per defecte `zeros`.
2. `g = ordfilt2(f,25,ones(5),symmetric)`. És un filtre de màxims que agafa el valor més gran dels de la màscara. Finestra 5x5 i opcions de `padding` simètriques.

`colfilt`

El filtre transforma la imatge per seccions en columnes per guanyar velocitat al moment del filtratge i aplica una funció de filtrat. La sintaxi d'aquest filtre és:

```
g = colfilt(f,[m n],block_type,fun)
```

on `g` és la imatge final, `f` és la imatge inicial, `[m n]` és la mida del filtre que serà tractat com a columna, `block_type` determina la forma en que es formen les columnes i son filtrades i `fun` és la funció aplicada per el filtratge.

- `block_type` pot fer dues referències:
  1. `distinct`: Cada bloc mxn distint és tractat com una matriu columna i filtrat abans de substituir els valors de tota la secció mxn.
  2. `sliding`: Els blocs mxn transformats en columnes serveixen per tractar un únic valor. Després es substitueix el valor del píxel d'estudi i es desplaça la màscara mxn al següent píxel i es repeteix l'operació.
- `fun` pot ser una funció aritmètica o no, com `@mean` (mitjana), `@min` (mínim) o `@max` (màxim).

`imfilter`

És una funció de filtratge amb moltes funcionalitats. Incorpora opcions de filtratge, de contorn i de tamany. La funció es crida de la següent forma:

```
g = imfilter(f,w,filtering_mode,boundary_options,size_options)
```

on `g` és la imatge final, `f` és la imatge d'entrada, `w` és el filtre a aplicar, `filtering_mode` defineix el mode de filtratge (correlació o convolució), `boundary_options` aplica les opcions de contorn (`padding`) i `size_options` determina la mida final de la imatge (`same` per dimensió igual a la d'entrada i `full` per dimensió total juntament amb les opcions de contorn).

Les opcions per defecte son: correlació, contorn ampliat amb zeros i opcions de tamany `same`.

Aquest filtre sol ser emprat juntament amb la funció fspecial, que serveix per crear filtres complexes com son: filtre gaussià, filtre laplacià, filtre de Sobel, filtre de moviment o filtre log (laplacià d'un gaussià).

Un exemple d'aquest filtre podria ser:

```
w = fspecial(gaussian,5,0.5);

g = imfilter(f,w,symmetric,full);
```

Aquestes instruccions passen la imatge per un filtre gaussià de mida 5x5 i desviació típica 0.5. Les opcions de contorn seran simètriques i el tamany final de la imatge inclourà els valors emprats per les opcions de contorn.

`medfilt2`

Es tracta d'una funció exclusiva de filtre de medianes per imatges d'intensitat. Per tant, aquesta funció retorna una imatge passada per un filtre de medianes personalitzat per els paràmetres de la funció.

La funció es crida de la següent manera:

```
g = medfilt2(f,[m  n],padopt)
```

on `g` és la imatge de sortida, `f` la imatge d'entrada, `m` i `n` defineixen les dimensions de la màscara del filtre (files i columnes) i `padopt` determina les opcions de contorn.

El filtre de medianes és un filtre molt eficaç per llevar renou sense donar aspecte desenfocat de més a la imatge.

Un exemple d'aplicació seria:

```
g = medfilt2(f,[5  5],symmetric);
```

Passa la imatge de intensitat en escala de grisos per un filtre de medianes 5x5 i condicions de contorn `symmetric`

Existeixen altres opcions de filtrat de matrius a MATLAB, però es tracten de variacions dels anomenats i no son necessaris per entendre el funcionament de l'algoritme ni els principis de filtrat digital.

APÈNDIX D

ALGORITME DE PROCESSAT PRINCIPAL

```
m1 = 9;
r = 10;
a = 5;
t = 0.25;
m2 = 5;

f = imread('FLIR0094.jpg');

% f_R = f(:,:,1);
% f_G = f(:,:,2);
% f_B = f(:,:,3);

% g_R = medfilt2(f_R, 'symmetric', [m1 m1]);
% g_G = medfilt2(f_G, 'symmetric', [m1 m1]);
% g_B = medfilt2(f_B, 'symmetric', [m1 m1]);
%
% g(:,:,1) = g_R;
% g(:,:,2) = g_G;
% g(:,:,3) = g_B;

g = RGBmedfilt(f, m1);

h = imsharpen(g, 'radius', r, 'amount', a);

hGray = rgb2gray(h);

%t = graythresh(hGray) %Activar la línia per threshold automàtic

hBin = im2bw(hGray,t);

hBinFill = complementar(hBin);

hBinFill = imfill(hBinFill);

```

```
35 hBinFill = complementar(hBinFill);
36
37 iBin = imclose(¬hBin, strel('disk', 3));
38 iBin = ¬iBin;
39
40 hBinMed = medfilt2(hBin, 'symmetric', [m2 m2]);
41
42 hBinFillMed = medfilt2(hBinFill, 'symmetric', [m2 m2]);
43
44 iBinMed = medfilt2(iBin, 'symmetric', [m2 m2]);
45
46 hBinEdge = edge(hBinMed, 'canny');
47
48 iBinEdge = edge(iBinMed, 'canny');
49
50 j = remarcar(f, hBinMed, 0, 255, 0);
51
52 k = remarcar(f, iBinMed, 0, 255, 0);
53
54 l = remarcar(f, iBinEdge, 0, 255, 0);
55
56 % subplot(3,5,1);
57 imshow(f);
58 title('Original');
59 % subplot(3,5,2);
60 % imshow(g);
61 % imshow(hGray);
62 % title('Filtre mediana 9x9');
63 % subplot(3,5,3);
64 % imshow(h);
65 % title('Sharpen r = 10, a = 5');
66 % subplot(3,5,4);
67 % imshow(hBin);
68 % title('Binarització');
69 % subplot(3,5,5);
70 % imshow(hBinFill);
71 % title('imfill amb imatge invertida');
72 % subplot(3,5,6);
73 % imshow(iBin);
74 % title('imclose (disk r=3) amb imatge invertida');
75 % subplot(3,5,7);
76 % imshow(hBinMed);
77 % title('Filtre mediana 5x5 (no imfill)');
78 % subplot(3,5,8);
79 % imshow(hBinFillMed);
80 % title('Filtre mediana 5x5 (amb imfill)');
81 % subplot(3,5,9);
82 % imshow(iBinMed);
83 % title('Filtre mediana 5x5 (amb imclose)');
84 % subplot(3,5,10);
85 % imshow(hBinEdge);
86 % title('Edge filtre mediana 5x5 (no imfill)');
87 % subplot(3,5,11);
88 % imshow(iBinEdge);
89 % title('Edge filtre mediana 5x5 (amb imclose)');
90 % subplot(3,5,12);
91 % imshow(j);
```

```
92 % title('Filtre mediana 5x5 (no imfill) remarcat');
93 % subplot(3,5,13);
94 % imshow(k);
95 % title('Filtre mediana 5x5 (amb imclose) remarcat');
96 % subplot(3,5,15);
97 % imshow(l);
98 % title('Edge filtre mediana 5x5 (amb imclose) remarcat');
```

APÈNDIX E

FUNCIÓ RGBMEDFILT

```
1  function f = RGBmedfilt(varargin)
2  %RGBMEDFILT(G, S, W) La imatge RGB G es passa per un filtre de ...
       medianes
3  %amb opcions de padding s i tamany W. En cas de no introduir ...
       valors per s
4  %i W s'empraràn valors 'symmetric' i '3x3' respectivament.
5  %Per ajuda de medfilt2 escriure 'help medfilt2'
6
7  g = varargin{1};
8
9  %Comprobació inicial
10
11 d = size(g);
12 d = [d 0 0];
13 if(¬strcmp(class(g), 'uint8') && ¬strcmp(class(g), 'double') || ...
       d(1,3) ≠ 3)
14     error('Incorrect image "g" format.')
15 end
16
17 %Comprobació varargin
18
19 if nargin == 1
20     s = 'symmetric';
21     W = 3;
22
23 elseif nargin == 2
24     if strcmp(class(varargin{2}), 'char')
25         s = varargin{2};
26         W = 3;
27     elseif strcmp(class(varargin{2}), 'double')
28         s = 'symmetric';
29         W = varargin{2};
30     else
31         error('Input arguments do not match with argument types.')
```

```
32      end
33
34  elseif nargin == 3
35      if strcmp(class(varargin{2}), 'char')
36          s = varargin{2};
37          W = varargin{3};
38      elseif strcmp(class(varargin{2}), 'double')
39          s = varargin{3};
40          W = varargin{2};
41      else
42          error('Input arguments do not match with argument types.')
43      end
44
45  else
46      error('Too many input arguments.');
47  end
48
49  %Inici
50
51  g_R = g(:,:,1);
52  g_G = g(:,:,2);
53  g_B = g(:,:,3);
54
55  f_R = medfilt2(g_R, s, [W W]);
56  f_G = medfilt2(g_G, s, [W W]);
57  f_B = medfilt2(g_B, s, [W W]);
58
59  f(:,:,1) = f_R;
60  f(:,:,2) = f_G;
61  f(:,:,3) = f_B;
62
63  end
```

APÈNDIX F

FUNCIÓ COMPLEMENTAR

```
1  function f = complementar(g)
2  %COMPLEMENTAR(G) guarda a "f" els valors complementaris de "g"
3  %    F = COMPLEMENTAR(G) Crea la imatge F amb els valors ...
       complementaris de G
4  %    Pot emprar-se per imatges en escala de grisos o imatges ...
       binàries.
5  %    F/G - imatges grises/binàries.
6
7  %Comprobació inicial
8
9  c = class(g);
10
11 if(strcmp(c, 'uint8'))
12     g = double(g)/255;
13 end
14
15 maxVal = max(max(g));
16 minVal = min(min(g));
17
18 if(minVal < 0 || maxVal > 1)
19     error('Incorrect "G" values.')
20 end
21
22 %Inici
23
24 [M, N] = size(g);
25
26 for i = 1:1:M
27     for j = 1:1:N
28         f(i,j) = 1 - g(i,j);
29     end
30 end
31
32 %Reconversió
```

```
33
34 if(strcmp(c, 'uint8'))
35     f = uint8(f*255);
36 end
```

APÈNDIX G

FUNCIÓ REMARCAR

```
1  function f = remarcar(g, h, R, G, B)
2  %REMARCAR Marca los valores altos de una imagen en otra.
3  %f = remarcar(g, h, R, G, B)
4  %   Remarca los valores altos(1s) de una imagen "h" en la imagen "g"
5  %   del color definido por los valores R, G y B, correspondientes
6  %   al código de colores RGB del color deseado.
7  %   h - imagen binaria | g/f - imagen RGB | R/G/B - uint8 [0-255]
8  %   Ejemplos R/G/B: 255, 0, 0 - Rojo | 0, 255, 0 - Verde |
9  %   0, 0, 255 - Azul | 255, 255, 255 - Blanco | 0, 0, 0 - Negro
10
11 %Comprobación inicial
12
13 if (R < 0 || G < 0 || B < 0 || R > 255 || G > 255 || B > 255)
14     error('R/G/B values must be unsigned 8-bit integers [0-255].')
15 end
16
17 d = size(g);
18 d = [d 0];
19 if (¬strcmp(class(g), 'uint8') && ¬strcmp(class(g), 'double') || ...
       d(1,3) ≠ 3)
20     error('Incorrect image "g" format.')
21 end
22
23 if strcmp(class(g), 'double')
24    R = R/255;
25    G = G/255;
26    B = B/255;
27 end
28
29 d = size(h);
30 d = size(d);
31
32 if (¬strcmp(class(h), 'logical') || d(1,2) ≠ 2)
33     error('Incorrect image "h" format.')
```

```
34 end
35
36 %Configuarción inicial
37
38 g_R = g(:,:,1);
39 g_G = g(:,:,2);
40 g_B = g(:,:,3);
41
42 [M, N] = size(g_R);
43
44 f_R = g_R;
45 f_G = g_G;
46 f_B = g_B;
47
48 %Comprobación tamaño
49
50 if (size(g_R) ≠ size(h))
51     error('size of "g" and "h" do not match.')
52 end
53
54 %Inicio función
55
56 for i = 1:1:M
57     for j = 1:1:N
58         if (h(i,j) == 1)
59             f_R(i,j) = R;
60             f_G(i,j) = G;
61             f_B(i,j) = B;
62         end
63     end
64 end
65
66 f(:,:,1) = f_R;
67 f(:,:,2) = f_G;
68 f(:,:,3) = f_B;
69
70 end
```

APÈNDIX H

FUNCIÓ ALGORITMEGUI

```
1  function g = algoritmeGUI(f, temp, med1, shar, bin, fill, clo, ...
       med2, edg, rem, m1, r, a, t, m2)
2
3  %      figure;
4  %      subplot(3,3,1);
5  %      imshow(f);
6
7      if ¬(med1||shar||bin||med2)
8          g = f;
9      else
10         h = f;
11         if med1
12             h = RGBmedfilt(f, m1);
13
14 %             subplot(3,3,2);
15 %             imshow(h);
16         end
17
18         if shar
19             h = imsharpen(h, 'radius', r, 'amount', a);
20
21 %             subplot(3,3,3);
22 %             imshow(h);
23         end
24
25         if bin
26             hGray = rgb2gray(h);
27             if t == 2
28                 t = graythresh(hGray);
29             end
30
31             h = im2bw(hGray, t);
32
33             if temp
```

```
34
35                  h = ¬h;
36              end
37
38  %             subplot(3,3,4);
39  %             imshow(h);
40          end
41
42          if fill
43              hCom = complementar(h);
44
45              hFill = imfill(hCom);
46
47              h = complementar(hFill);
48
49  %             subplot(3,3,5);
50  %             imshow(h);
51          end
52
53          if clo
54              hClo = imclose(¬h, strel('disk', 3));
55              h = ¬hClo;
56
57  %             subplot(3,3,5);
58  %             imshow(h);
59          end
60
61          if med2
62              h = medfilt2(h, 'symmetric', [m2 m2]);
63
64  %             subplot(3,3,6);
65  %             imshow(h);
66          end
67
68          if edg
69              h = edge(h, 'canny');
70
71  %             subplot(3,3,7);
72  %             imshow(h);
73          end
74
75          if rem
76              h = remarcar(f, h, 0, 255, 0);
77
78  %             subplot(3,3,8);
79  %             imshow(h);
80          end
81
82          g = h;
83      end
84
85  end
```

APÈNDIX A

CODI GUI

```
function varargout = GUI_Imatges(varargin)
% GUI_IMATGES MATLAB code for GUI_Imatges.fig
%      GUI_IMATGES, by itself, creates a new GUI_IMATGES or ... raises the existing
%      singleton*.
%
%      H = GUI_IMATGES returns the handle to a new GUI_IMATGES ... or the handle to
%      the existing singleton*.
%
%      GUI_IMATGES('CALLBACK',hObject,eventData,handles,...) ... calls the local
%      function named CALLBACK in GUI_IMATGES.M with the given ... input arguments.
%
%      GUI_IMATGES('Property','Value',...) creates a new ... GUI_IMATGES or raises the
%      existing singleton*.  Starting from the left, property ... value pairs are
%      applied to the GUI before GUI_Imatges_OpeningFcn gets ... called.  An
%      unrecognized property name or invalid value makes ... property application
%      stop.  All inputs are passed to GUI_Imatges_OpeningFcn ... via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI ... allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI_Imatges

```

```
25 % Last Modified by GUIDE v2.5 17-May-2017 11:22:24
26
27 % Begin initialization code - DO NOT EDIT
28 gui_Singleton = 1;
29 gui_State = struct('gui_Name',       mfilename, ...
30                    'gui_Singleton',  gui_Singleton, ...
31                    'gui_OpeningFcn', @GUI_Imatges_OpeningFcn, ...
32                    'gui_OutputFcn',  @GUI_Imatges_OutputFcn, ...
33                    'gui_LayoutFcn',  [] , ...
34                    'gui_Callback',   []);
35 if nargin && ischar(varargin{1})
36     gui_State.gui_Callback = str2func(varargin{1});
37 end
38
39 if nargout
40     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
41 else
42     gui_mainfcn(gui_State, varargin{:});
43 end
44 % End initialization code - DO NOT EDIT
45 end
46
47 % --- Executes just before GUI_Imatges is made visible.
48 function GUI_Imatges_OpeningFcn(hObject, eventdata, handles, ...
       varargin)
49 % This function has no output args, see OutputFcn.
50 % hObject    handle to figure
51 % eventdata  reserved - to be defined in a future version of MATLAB
52 % handles    structure with handles and user data (see GUIDATA)
53 % varargin   command line arguments to GUI_Imatges (see VARARGIN)
54
55 % Choose default command line output for GUI_Imatges
56 handles.output = hObject;
57
58 % Update handles structure
59 guidata(hObject, handles);
60
61 % UIWAIT makes GUI_Imatges wait for user response (see UIRESUME)
62 % uiwait(handles.figure1);
63
64 setGlobalm1(9);
65 setGlobalr(10);
66 setGlobala(5);
67 setGlobalt(2);
68 setGlobalm2(5);
69 setGlobalmed1(false);
70 setGlobalshar(false);
71 setGlobalbin(false);
72 setGlobalfill(false);
73 setGlobalclo(false);
74 setGlobalmed2(false);
75 setGlobaledg(false);
76 setGlobalrem(false);
77 setGlobaltemp(false);
78
79 end
80
```

```
81 % --- Outputs from this function are returned to the command line.
82 function varargout = GUI_Imatges_OutputFcn(hObject, eventdata, ...
       handles)
83 % varargout  cell array for returning output args (see VARARGOUT);
84 % hObject    handle to figure
85 % eventdata  reserved - to be defined in a future version of MATLAB
86 % handles    structure with handles and user data (see GUIDATA)
87
88 % Get default command line output from handles structure
89 varargout{1} = handles.output;
90 end
91
92 function editMedSize_Callback(hObject, eventdata, handles)
93 % hObject    handle to editMedSize (see GCBO)
94 % eventdata  reserved - to be defined in a future version of MATLAB
95 % handles    structure with handles and user data (see GUIDATA)
96
97 % Hints: get(hObject,'String') returns contents of editMedSize ...
       as text
98 %        str2double(get(hObject,'String')) returns contents of ...
       editMedSize as a double
99
100     setGlobalm1(str2double(get(hObject,'String')));
101     if isempty(get(hObject,'String'))
102         setGlobalm1(9);
103     end
104
105 end
106
107 function editRadius_Callback(hObject, eventdata, handles)
108 % hObject    handle to edit2 (see GCBO)
109 % eventdata  reserved - to be defined in a future version of MATLAB
110 % handles    structure with handles and user data (see GUIDATA)
111
112 % Hints: get(hObject,'String') returns contents of edit2 as text
113 %        str2double(get(hObject,'String')) returns contents of ...
       edit2 as a double
114
115
116     setGlobalr(str2double(get(hObject, 'String')));
117     if isempty(get(hObject,'String'))
118         setGlobalr(10);
119     end
120
121 end
122
123 function editAmount_Callback(hObject, eventdata, handles)
124 % hObject    handle to edit4 (see GCBO)
125 % eventdata  reserved - to be defined in a future version of MATLAB
126 % handles    structure with handles and user data (see GUIDATA)
127
128 % Hints: get(hObject,'String') returns contents of edit4 as text
129 %        str2double(get(hObject,'String')) returns contents of ...
       edit4 as a double
130
131     setGlobala(str2double(get(hObject,'String')));
132     if isempty(get(hObject,'String'))
```

```
133         setGlobala(5);
134     end
135
136 end
137
138 % --- Executes on button press in pushProces.
139 function pushProces_Callback(hObject, eventdata, handles)
140 % hObject    handle to pushProces (see GCBO)
141 % eventdata  reserved - to be defined in a future version of MATLAB
142 % handles    structure with handles and user data (see GUIDATA)
143
144     [file, lDir] = uigetfile(...
145         {'*.jpg; *.JPG; *.jpeg; *.JPEG; *.img; *.IMG; *.tif; ...
                *.TIF; *.tiff; *.TIFF','Supported Files ...
                (*.jpg,*.img,*.tiff,*.tif)'; ...
146         '*.jpg','jpg Files (*.jpg)';...
147         '*.JPG','JPG Files (*.JPG)';...
148         '*.jpeg','jpeg Files (*.jpeg)';...
149         '*.JPEG','JPEG Files (*.JPEG)';...
150         '*.img','img Files (*.img)';...
151         '*.IMG','IMG Files (*.IMG)';...
152         '*.tif','tif Files (*.tif)';...
153         '*.TIF','TIF Files (*.TIF)';...
154         '*.tiff','tiff Files (*.tiff)';...
155         '*.TIFF','TIFF Files (*.TIFF)'},...
156         'MultiSelect', 'on');
157
158     if ¬(isnumeric(file) || isnumeric(lDir))
159         sDir = uigetdir('Desar imatges a...');
160
161         if ¬isnumeric(file)
162             med1 = getGlobalmed1;
163             shar = getGlobalshar;
164             bin = getGlobalbin;
165             fill = getGlobalfill;
166             clo = getGlobalclo;
167             med2 = getGlobalmed2;
168             edg = getGlobaledg;
169             rem = getGlobalrem;
170             temp = getGlobaltemp;
171             m1 = getGlobalm1;
172             r = getGlobalr;
173             a = getGlobala;
174             t = getGlobalt;
175             m2 = getGlobalm2;
176
177             oldDir = cd(lDir);
178
179             i = size(file);
180
181             for j = 1:1:i(2)
182                 d = file{j};
183                 f = imread(d);
184
185                 g = algoritmeGUI(f, temp, med1, shar, bin, fill, ...
                        clo, med2, edg, rem, m1, r, a, t, m2);
186
```

```
                    name = ['Processed_' d];

                    cd(sDir);
                    imwrite(g, name);
                    cd(oldDir);
                end
            end
        end

end

% --- Executes on button press in pushProva.
function pushProva_Callback(hObject, eventdata, handles)
% hObject    handle to pushProva (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

f = getGlobalf;
med1 = getGlobalmed1;
shar = getGlobalshar;
bin = getGlobalbin;
fill = getGlobalfill;
clo = getGlobalclo;
med2 = getGlobalmed2;
edg = getGlobaledg;
rem = getGlobalrem;
temp = getGlobaltemp;
m1 = getGlobalm1;
r = getGlobalr;
a = getGlobala;
t = getGlobalt;
m2 = getGlobalm2;

g = algoritmeGUI(f, temp, med1, shar, bin, fill, clo, med2, edg, ...
    rem, m1, r, a, t, m2);
setGlobalg(g);

axes(getGlobalAxes);
imshow(g);

end

% --- Executes on button press in pushBrowse.
function pushBrowse_Callback(hObject, eventdata, handles)
% hObject    handle to pushBrowse (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[file, directory] = uigetfile(...
    {'*.jpg; *.JPG; *.jpeg; *.JPEG; *.img; *.IMG; *.tif; *.TIF; ...
        *.tiff, *.TIFF','Supported Files ...
        (*.jpg,*.img,*.tiff,*.tif)'; ...
    '*.jpg','jpg Files (*.jpg)';...
    '*.JPG','JPG Files (*.JPG)';...
    '*.jpeg','jpeg Files (*.jpeg)';...
    '*.JPEG','JPEG Files (*.JPEG)';...
    '*.img','img Files (*.img)';...
    '*.IMG','IMG Files (*.IMG)';...
```

```
241      '*.tif','tif Files (*.tif)';...
242      '*.TIF','TIF Files (*.TIF)';...
243      '*.tiff','tiff Files (*.tiff)';...
244      '*.TIFF','TIFF Files (*.TIFF)'},...
245      'MultiSelect', 'off');
246
247      setGlobalFile(file);
248      setGlobalDir(directory);
249
250      str = [directory file];
251
252      if ¬(isnumeric(directory) || isnumeric(file))
253          i = imread(str);
254          setGlobalf(i);
255
256          h = findobj('Type', 'axes');
257          setGlobalAxes(h);
258
259          axes(h);
260          imshow(i);
261
262          h = findobj('Style', 'pushbutton');
263          set(h, 'Enable', 'on');
264      end
265  end
266
267  % --- Executes on button press in checkMed.
268  function checkMed_Callback(hObject, eventdata, handles)
269  % hObject    handle to checkMed (see GCBO)
270  % eventdata  reserved - to be defined in a future version of MATLAB
271  % handles    structure with handles and user data (see GUIDATA)
272
273  % Hint: get(hObject,'Value') returns toggle state of checkMed
274
275      setGlobalmed1(get(hObject,'Value'));
276      h = findobj('tag', 'editMedSize');
277
278      if get(hObject,'Value')
279          set(h, 'visible', 'on')
280      else
281          set(h, 'visible', 'off')
282      end
283
284  end
285
286  % --- Executes on button press in checkSharpen.
287  function checkSharpen_Callback(hObject, eventdata, handles)
288  % hObject    handle to checkSharpen (see GCBO)
289  % eventdata  reserved - to be defined in a future version of MATLAB
290  % handles    structure with handles and user data (see GUIDATA)
291
292  % Hint: get(hObject,'Value') returns toggle state of checkSharpen
293
294      setGlobalshar(get(hObject,'Value'));
295      h1 = findobj('tag', 'editRadius');
296      h2 = findobj('tag', 'editAmount');
297
```

```
298     if get(hObject,'Value')
299         set(h1, 'visible', 'on')
300         set(h2, 'visible', 'on')
301     else
302         set(h1, 'visible', 'off')
303         set(h2, 'visible', 'off')
304     end
305 end
306 
307 
308 % --- Executes on button press in checkFill.
309 function checkFill_Callback(hObject, eventdata, handles)
310 % hObject    handle to checkFill (see GCBO)
311 % eventdata  reserved - to be defined in a future version of MATLAB
312 % handles    structure with handles and user data (see GUIDATA)
313 
314 % Hint: get(hObject,'Value') returns toggle state of checkFill
315 
316     setGlobalfill(get(hObject,'Value'));
317     setGlobalclo(¬get(hObject,'Value'));
318 
319     if get(hObject,'Value')
320         h=findobj('tag','checkClose');
321         set(h,'value',0);
322         set(gcbo, 'value', 1);
323     end
324 
325 end
326 
327 
328 function editThresh_Callback(hObject, eventdata, handles)
329 % hObject    handle to editThresh (see GCBO)
330 % eventdata  reserved - to be defined in a future version of MATLAB
331 % handles    structure with handles and user data (see GUIDATA)
332 
333 % Hints: get(hObject,'String') returns contents of editThresh as ...
        text
334 %        str2double(get(hObject,'String')) returns contents of ...
        editThresh as a double
335 
336     setGlobalt(str2double(get(hObject,'String')));
337     if isempty(get(hObject,'String'))
338         setGlobalt(2);
339     end
340 
341 end
342 
343 % --- Executes on button press in checkBin.
344 function checkBin_Callback(hObject, eventdata, handles)
345 % hObject    handle to checkBin (see GCBO)
346 % eventdata  reserved - to be defined in a future version of MATLAB
347 % handles    structure with handles and user data (see GUIDATA)
348 
349 % Hint: get(hObject,'Value') returns toggle state of checkBin
350 
351     setGlobalbin(get(hObject,'Value'));
352     h=findobj('tag','editThresh');
```

```
353      h1 = findobj('tag', 'checkFill');
354      h2 = findobj('tag', 'checkClose');
355      h3 = findobj('tag', 'checkEdge');
356      h4 = findobj('tag', 'checkRemarcar');
357
358      if get(hObject,'Value')
359          set(h, 'visible', 'on');
360          set(h1, 'visible', 'on');
361          set(h2, 'visible', 'on');
362          set(h3, 'visible', 'on');
363          set(h4, 'visible', 'on');
364
365      else
366          set(h, 'visible', 'off');
367          set(h1, 'visible', 'off', 'value', 0);
368          setGlobalfill(0);
369          set(h2, 'visible', 'off', 'value', 0);
370          setGlobalclo(0);
371          set(h3, 'visible', 'off', 'value', 0);
372          setGlobalrem(0);
373          set(h4, 'visible', 'off', 'value', 0);
374      end
375
376  end
377
378  % --- Executes on button press in checkClose.
379  function checkClose_Callback(hObject, eventdata, handles)
380  % hObject    handle to checkClose (see GCBO)
381  % eventdata  reserved - to be defined in a future version of MATLAB
382  % handles    structure with handles and user data (see GUIDATA)
383
384  % Hint: get(hObject,'Value') returns toggle state of checkClose
385
386      setGlobalclo(get(hObject,'Value'));
387      setGlobalfill(¬get(hObject,'Value'));
388
389      if get(hObject,'Value')
390          h=findobj('tag','checkFill');
391          set(h,'value',0);
392          set(gcbo, 'value', 1);
393      end
394
395  end
396
397  % --- Executes on button press in checkMed1.
398  function checkMed1_Callback(hObject, eventdata, handles)
399  % hObject    handle to checkMed1 (see GCBO)
400  % eventdata  reserved - to be defined in a future version of MATLAB
401  % handles    structure with handles and user data (see GUIDATA)
402
403  % Hint: get(hObject,'Value') returns toggle state of checkMed1
404
405      setGlobalmed2(get(hObject,'Value'));
406      h = findobj('tag', 'editMedSize1');
407
408      if get(hObject,'Value')
409          set(h, 'visible', 'on')
```

```
410     else
411         set(h, 'visible', 'off')
412     end
413 end
414 
415 
416 function editMedSize1_Callback(hObject, eventdata, handles)
417 % hObject    handle to editMedSize1 (see GCBO)
418 % eventdata  reserved - to be defined in a future version of MATLAB
419 % handles    structure with handles and user data (see GUIDATA)
420 
421 % Hints: get(hObject,'String') returns contents of editMedSize1 ...
        as text
422 %        str2double(get(hObject,'String')) returns contents of ...
        editMedSize1 as a double
423 
424     setGlobalm2(str2double(get(hObject,'String')));
425     if isempty(get(hObject,'String'))
426         setGlobalm2(5);
427     end
428 
429 end
430 
431 % --- Executes on button press in pushDesar.
432 function pushDesar_Callback(hObject, eventdata, handles)
433 % hObject    handle to pushDesar (see GCBO)
434 % eventdata  reserved - to be defined in a future version of MATLAB
435 % handles    structure with handles and user data (see GUIDATA)
436 
437     [file, path] = uiputfile('*.jpg', 'Desar Imatge com...');
438 
439     if ¬(isnumeric(path) || isnumeric(file))
440         oldDir = cd(path);
441         imwrite(getGlobalg, file);
442         cd(oldDir);
443     end
444 
445 end
446 
447 % --- Executes on button press in checkRemarcar.
448 function checkRemarcar_Callback(hObject, eventdata, handles)
449 % hObject    handle to checkRemarcar (see GCBO)
450 % eventdata  reserved - to be defined in a future version of MATLAB
451 % handles    structure with handles and user data (see GUIDATA)
452 
453 % Hint: get(hObject,'Value') returns toggle state of checkRemarcar
454 
455     setGlobalrem(get(hObject,'Value'));
456 
457 end
458 
459 % --- Executes on button press in checkEdge.
460 function checkEdge_Callback(hObject, eventdata, handles)
461 % hObject    handle to checkEdge (see GCBO)
462 % eventdata  reserved - to be defined in a future version of MATLAB
463 % handles    structure with handles and user data (see GUIDATA)
464 
```

```
465 % Hint: get(hObject,'Value') returns toggle state of checkEdge
466
467     setGlobaledg(get(hObject,'Value'));
468
469 end
470
471 % --- Executes on button press in checkTemp.
472 function checkTemp_Callback(hObject, eventdata, handles)
473 % hObject    handle to checkTemp (see GCBO)
474 % eventdata  reserved - to be defined in a future version of MATLAB
475 % handles    structure with handles and user data (see GUIDATA)
476
477 % Hint: get(hObject,'Value') returns toggle state of checkTemp
478
479     setGlobaltemp(get(hObject,'Value'));
480
481 end
482
483 % --- Executes on button press in pushReset.
484 function pushReset_Callback(hObject, eventdata, handles)
485 % hObject    handle to pushReset (see GCBO)
486 % eventdata  reserved - to be defined in a future version of MATLAB
487 % handles    structure with handles and user data (see GUIDATA)
488
489     h = findobj(gcbf, 'Style', 'checkbox');
490     set(h, 'Value', 0);
491
492     setGlobalmed1(0);
493     setGlobalshar(0);
494     setGlobalbin(0);
495     setGlobalfill(0);
496     setGlobalclo(0);
497     setGlobalmed2(0);
498     setGlobaledg(0);
499     setGlobalrem(0);
500     setGlobaltemp(0);
501
502     h = findobj('tag', 'editMedSize');
503     set(h, 'String', 9)
504     setGlobalm1(9);
505
506     h = findobj('tag', 'editRadius');
507     set(h, 'String', 10)
508     setGlobalr(10);
509
510     h = findobj('tag', 'editAmount');
511     set(h, 'String', 5)
512     setGlobala(5);
513
514     h = findobj('tag', 'editThresh');
515     set(h, 'String', 'Auto')
516     setGlobalt(2);
517
518     h = findobj('tag', 'editMedSize1');
519     set(h, 'String', 5)
520     setGlobalm2(5);
521
```

```
522     axes(getGlobalAxes);
523     imshow(getGlobalf);
524 
525 end
526 
527 % --- Funcions de tractament de variables globals
528 function setGlobalr(val)
529 global r;
530 r = val;
531 end
532 
533 function x = getGlobalr
534 global r;
535 x = r;
536 end
537 
538 function setGlobala(val)
539 global a;
540 a = val;
541 end
542 
543 function x = getGlobala
544 global a;
545 x = a;
546 end
547 
548 function setGlobalt(val)
549 global t;
550 t = val;
551 end
552 
553 function x = getGlobalt
554 global t;
555 x = t;
556 end
557 
558 function setGlobalm1(val)
559 global m1;
560 m1 = val;
561 end
562 
563 function x = getGlobalm1
564 global m1;
565 x = m1;
566 end
567 
568 function setGlobalm2(val)
569 global m2;
570 m2 = val;
571 end
572 
573 function x = getGlobalm2
574 global m2;
575 x = m2;
576 end
577 
578 function setGlobalf(img)
```

```
579 global f;
580 f = img;
581 end
582
583 function x = getGlobalf
584 global f;
585 x = f;
586 end
587
588 function setGlobalg(img)
589 global g;
590 g = img;
591 end
592
593 function x = getGlobalg
594 global g;
595 x = g;
596 end
597
598 function setGlobalAxes(val)
599 global axes;
600 axes = val;
601 end
602
603 function x = getGlobalAxes
604 global axes;
605 x = axes;
606 end
607
608 function setGlobalFile(val)
609 global file;
610 file = val;
611 end
612
613 function x = getGlobalFile
614 global file;
615 x = file;
616 end
617
618 function setGlobalDir(val)
619 global dir;
620 dir = val;
621 end
622
623 function x = getGlobalDir
624 global dir;
625 x = dir;
626 end
627
628 function setGlobalmed1(val)
629 global med1;
630 med1 = val;
631 end
632
633 function x = getGlobalmed1
634 global med1;
635 x = med1;
```

```
636 end
637
638 function setGlobalshar(val)
639 global shar;
640 shar = val;
641 end
642
643 function x = getGlobalshar
644 global shar;
645 x = shar;
646 end
647
648 function setGlobalbin(val)
649 global bin;
650 bin = val;
651 end
652
653 function x = getGlobalbin
654 global bin;
655 x = bin;
656 end
657
658 function setGlobalfill(val)
659 global fill;
660 fill = val;
661 end
662
663 function x = getGlobalfill
664 global fill;
665 x = fill;
666 end
667
668 function setGlobalclo(val)
669 global clo;
670 clo = val;
671 end
672
673 function x = getGlobalclo
674 global clo;
675 x = clo;
676 end
677
678 function setGlobalmed2(val)
679 global med2;
680 med2 = val;
681 end
682
683 function x = getGlobalmed2
684 global med2;
685 x = med2;
686 end
687
688 function setGlobalrem(val)
689 global rem;
690 rem = val;
691 end
692
```

```
693 function x = getGlobalrem
694 global rem;
695 x = rem;
696 end
697
698 function setGlobaledg(val)
699 global edg;
700 edg = val;
701 end
702
703 function x = getGlobaledg
704 global edg;
705 x = edg;
706 end
707
708 function setGlobaltemp(val)
709 global temp;
710 temp = val;
711 end
712
713 function x = getGlobaltemp
714 global temp;
715 x = temp;
716 end
```

APÈNDIX J

VISUALITZACIÓ DE PASSES I RESULTATS

A la següent imatge es representen diversos resultats després d'aplicar les diferentes passes de l'algoritme proposat a aquest treball així com la imatge final.

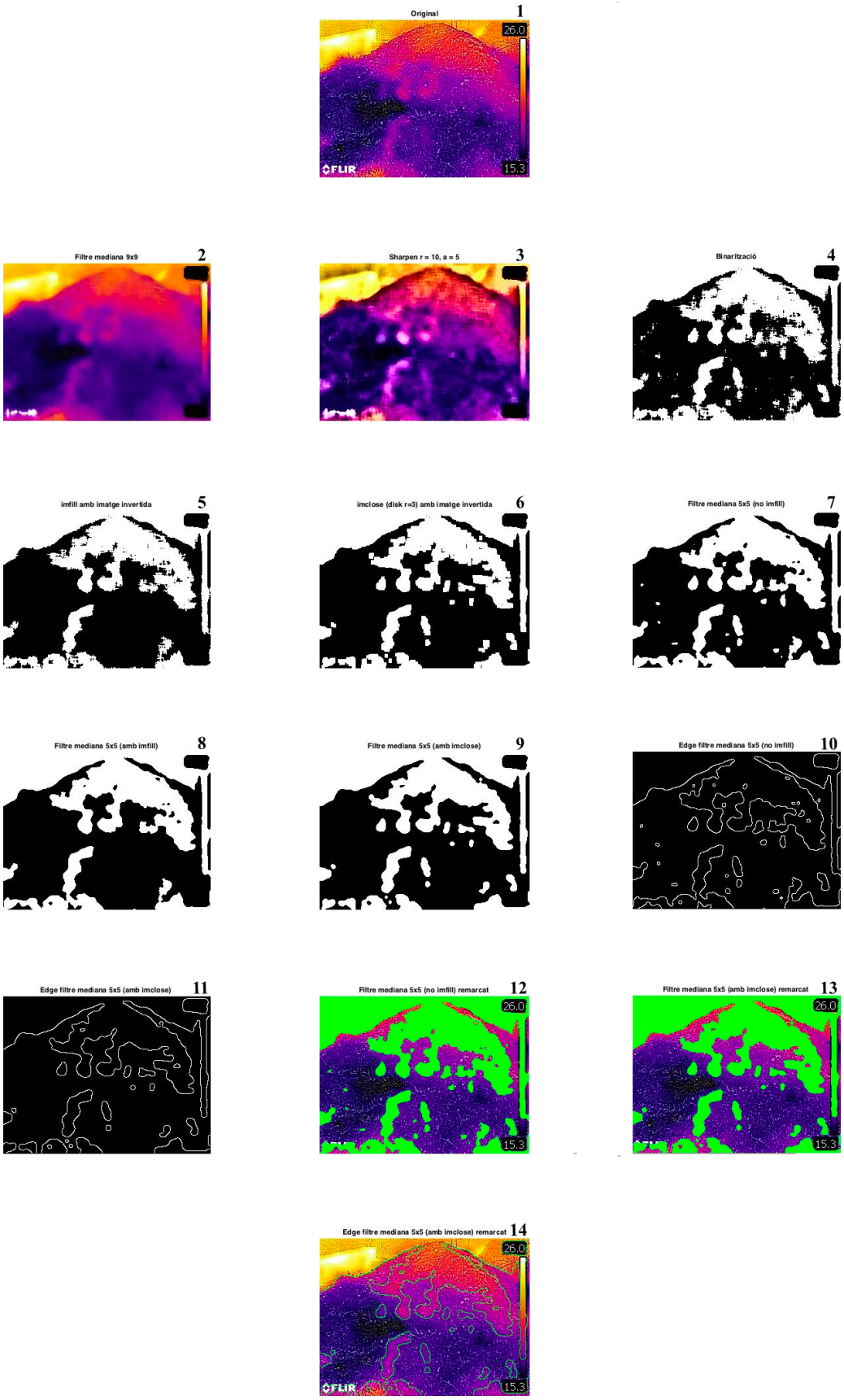

Figura J.1: Imatges resultants després de diferentes passes.

BIBLIOGRAFIA

[1] F. Tauro, S. Grimaldi, A. Petroselli, M. Rulli, and M. Porfiri, "Fluorescent particle tracers in surface hydrology: a proof of concept in a semi-natural hillslope," *Hydrology and Earth System Sciences*, 2012. 2.1

[2] F. Tauro, M. Aureli, M. Porfiri, and S. Grimaldi, "Characterization of buoyant fluorescent particles for field observations of water flows," *Sensors in Agriculture and Forestry*, 2010. 2.1

[3] A. Verma and R. Cooke, "Mapping subsurface drainage systems with color infrared aerial photographs," *AWRA Symposium on GIS and Water Resources*, 1996. 2.1

[4] D. J. Luscombe, K. Anderson, N. Gatis, E. Grand-Clement, and R. E. Brazier, "Using airborne thermal imaging data to measure near-surface hydrology in upland ecosystems." *Hidrological processes*, 2014. 2.1

[5] J. Proakis and D. Manolakis, *Tratamiento Digital de Señales*, 3rd ed. Prentice Hall, 1999. 2.2.1

[6] R. W. Dosselmann and A. Ye, "Lab: Intensity transformation and spatial filtering (http://www.cs.uregina.ca/Links/class-info/425-nova/Lab3/)."

[7] C. Wellons, "Linear spatial filters with gnu octave (http://nullprogram.com/blog/2008/02/22/)."

[8] T. Fletcher and Y. Gao, "Docencia: Image processing basics. spatial filtering (http://www.coe.utah.edu/~cs4640/slides/Lecture5.pdf)."

[9] F. A. Sarria, "Docencia: Fotointerpretación y teledetección. tema 1: Fundamentos físicos de la teledetección."

[10] C. S. Juan, "Docencia: Fonaments digital del senyal. capitol 3: Filtrado de imágenes (http://dmi.uib.es/~catalina/docencia/PDS/tema3.pdf)."

[11] "Mathworks (https://es.mathworks.com)."

[12] "Image processing research group (https://sites.google.com/site/internationalengineersrg/home)."

[13] R. E. González, R. E. Woods, and S. L. Eddins, *Digital Image Processing Using Matlab*, 2nd ed., 2009. C.2

[14] A. B. Biran, *What every engineer should know about Matlab and Simulink*. CRC Press, 2010.

[15] O. Marques, *Practical Image and Video Processing Using MATLAB*. Wiley, 2011.