Universitat de Vic
Escola Politècnica Superior

Treball Final de Màster

Programa d'anàlisi d'imatges de provetes metal·logràfiques

Antoni Suriñach i Albareda

Màster en Tecnologies Aplicades de la Informació

Director: Ramon Reig i Bolaño

Vic, setembre de 2010

Universitat de Vic
Escola Politècnica Superior

Resum del Treball Final de Màster
Màster en Tecnologies Aplicades de la Informació

Títol: Programa d'anàlisi d'imatges de provetes metal·lològràfiques
Paraules clau: instrumentació virtual, MATLAB, GUIDE, GUI, anàlisi d'imatges, metal·lografia, provetes metal·lològràfiques.
Autor: Antoni Suriñach i Albareda
Direcció: Ramon Reig i Bolaño
Data: 24 de setembre de 2010

Resum

L'objectiu del present TFM és explorar les possibilitats del programa matemàtic MATLAB i la seva eina Entorn de Disseny d'Interfícies Gràfiques d'Usuari (GUIDE), desenvolupant un programa d'anàlisi d'imatges de provetes metal·lològràfiques que es pugui utilitzar per a realitzar pràctiques de laboratori de l'assignatura Tecnologia de Materials de la titulació de Grau en Enginyeria Mecatrònica que s'imparteix a la Universitat de Vic.

Les àrees d'interès del treball són la Instrumentació Virtual, la programació MATLAB i les tècniques d'anàlisi d'imatges metal·lològràfiques.

En la memòria es posa un èmfasi especial en el disseny de la interfície i dels procediments per a efectuar les mesures. El resultat final és un programa que satisfà tots els requeriments que s'havien imposat en la proposta inicial. La interfície del programa és clara i neta, destinant molt espai a la imatge que s'analitza. L'estructura i disposició dels menús i dels comandaments ajuda a que la utilització del programa sigui fàcil i intuïtiva. El programa s'ha estructurat de manera que sigui fàcilment ampliable amb altres rutines de mesura, o amb l'automatització de les rutines existents.

Al tractar-se d'un programa que funciona com un instrument de mesura, es dedica un capítol sencer de la memòria a mostrar el procediment de càlcul dels errors que s'ocasionen durant la seva utilització, amb la finalitat de conèixer el seu ordre de magnitud, i de saber-los calcular de nou en cas que variïn les condicions d'utilització.

Pel que fa referència a la programació, malgrat que MATLAB no sigui un entorn de programació clàssic, sí que incorpora eines que permeten fer aplicacions no massa complexes, i orientades bàsicament a gràfics o a imatges. L'eina GUIDE simplifica la realització de la interfície d'usuari, malgrat que presenta problemes per tractar dissenys una mica complexos. Per altra banda, el codi generat per GUIDE no és accessible, cosa que no permet modificar manualment la interfície en aquells casos en els que GUIDE té problemes. Malgrat aquests petits problemes, la potència de càlcul de MATLAB compensa sobradament aquestes deficiències.

Universitat de Vic
Escola Politècnica Superior

Master Final Paper Abstract
Master in Applied Technologies of the Information

Title: software of image analysis of metallographic specimens
Key words: Virtual Instrumentation, MATLAB, GUIDE, GUI, image analysis, metallography, metallographic specimens.
Author: Antoni Suriñach i Albareda
Director: Ramon Reig i Bolaño
Date: 24 September 2010

Abstract

The aim of this Master Final Project is to explore the possibilities of the mathematical software MATLAB and its tool (GUIDE), developing a software of image analysis of metallographic specimens which can be used in laboratory practices of the subject Materials Technology given in the Mecatronic Engineering degree of the University of Vic.

The interest areas of the project are the Virtual Instrumentation, MATLAB programming and the analysis techniques of metallographic images.

In the written paper it is shown a special importance on the interface design and the procedures to execute the measures. The final result is a software that satisfies all the requirements imposed in the initial proposal. The software interface is clear and understandable, and very space is allocated to the analyzed image. The structure and layout of menus and commands help the software use to be easy and intuitive. The software is organized so that is easily expandable with other measure routines, or with the automation of the existing routines.

Being a program that works as a measuring instrument, a chapter of the written paper is dedicated to show the procedure for calculating the errors caused during its use. The purpose is to know its magnitude order and how to calculate them again if the utilization conditions have changed.

In terms of programming, although MATLAB is not a typical programming environment, it includes tools that allow applications not too complex and basically directed to graphics or images. The GUIDE tool simplifies the implementation of UI, although it presents some problems dealing with designs a bit complex. Furthermore, the code generated by GUIDE is not accessible, and it does not allow to modify manually the interface in cases of problems in GUIDE. Though these minor problems, the computing power of MATLAB amply compensates for these shortcomings.

Dedico aquest TFM a en Jaume i l'Adela. S'ho mereixen.

També el dedico a la Montse, la Laia, la Marta i la Bruna, en Jordi, en David i en Joel. Tots plegats estem compromesos en un projecte de vida engrescador, i ens il·lusionen i ens neguitegen les mateixes coses.

Agraeixo a en Ramon i la Judit tota la ajuda i el suport que m'han donat, i l'esforç que hi han esmerçat.

Finalment, agraeixo als professors, als companys de classe i als companys de feina l'estoïcitat i bon humor amb què han suportat les meves *dèries* d'aquest curs.

Índex

1. Introducció

El present Treball Final del *Màster en Tecnologies Aplicades de la Informació,* de la Universitat de Vic, titulat ***Programa d'anàlisi d'imatges de provetes metal·logràfiques (PRANIMET)***, és un projecte que s'inscriu en la família dels sistemes d'instrumentació virtual. Utilitza els recursos de programació gràfica i de càlcul matemàtic que ofereix l'entorn MATLAB per aconseguir un sistema de mesura, totalment gràfic, de determinades característiques de les imatges de provetes metal·logràfiques.

Així doncs, el projecte queda emmarcat en tres grans àmbits: per una banda, la Instrumentació Virtual; per l'altra, l'entorn de càlcul i programació MATLAB i, finalment, per les matèries de la Tecnologia de Materials.

1.1. Àmbit del treball

I La Instrumentació Virtual és la branca de la Instrumentació Electrònica que té com a objectiu la realització d'instruments de mesura mitjançant tècniques de programació informàtica, de manera que l'aparença i el maneig del programa resultant (el que s'anomena la interfície d'usuari) sigui molt semblant a la dels instruments tradicionals construïts amb dispositius electrònics.

II El programa MATLAB ®, de l'empresa MathWorks, és un llenguatge de computació tècnica d'alt nivell i un entorn interactiu per al desenvolupament d'algorismes, anàlisi i visualització de dades i càlcul numèric. La utilització del producte MATLAB pot resoldre problemes de computació tècnica més ràpidament que amb els llenguatges de programació tradicionals, com ara C, C++ i Fortran.

Es pot utilitzar MATLAB en una àmplia gamma d'aplicacions, incloent processament de senyals i imatges, comunicacions, disseny de control, prova i mesura, modelització i anàlisi financera, biologia computacional i altres. Una col·lecció de caixes d'eines complementàries (col·leccions de funcions de MATLAB per a propòsits especials, disponibles per separat) amplien l'entorn

de MATLAB per resoldre determinades classes de problemes en aquestes àrees d'aplicació.

Al mateix temps que totes les eines gràfiques i matemàtiques, MATLAB proporciona totes les característiques d'un llenguatge de programació tradicional, inclosos operadors aritmètics, control de flux, estructures de dades, programació orientada a objectes (POO), i característiques de depuració.

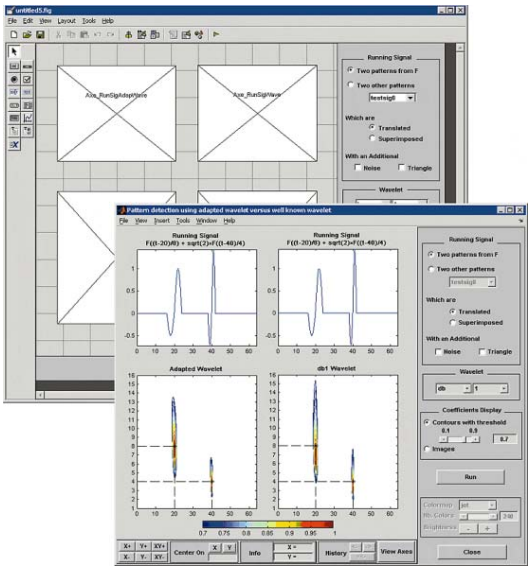


Figura 1.1. *Eina interactiva GUIDE de MATLAB.*

Mitjançant l'eina interactiva GUIDE (*Graphical User Interface Development Environment*) es poden dibuixar, dissenyar i editar interfícies d'usuari. GUIDE permet incloure quadres de llista, menús desplegables, botons, botons de radi i cursors, així com els gràfics de MATLAB i controls ActiveX. Com a alternativa a aquesta eina, també es poden crear interfícies gràfiques d'usuari mitjançant programació tradicional utilitzant les funcions de MATLAB.

III Els estudis de Grau en Enginyeria Mecatrònica que s'ofereixen a la Universitat de Vic treballen de manera interdisciplinària tècniques i coneixements de mecànica, electrònica, control i informàtica per tal de concebre noves maneres de produir, de desenvolupar nous productes i de dissenyar noves màquines.

En el seu Pla d'Estudis hi consta la matèria Tecnologia Mecànica, formada per 5 assignatures, entre les quals hi ha l'assignatura de **Tecnologia dels Materials** (1r curs, 2n quadrimestre, obligatòria, 6 crèdits).

Aquesta assignatura tracta de la relació entre la microestructura i les propietats dels materials per tal de poder tenir criteris per a la selecció i disseny de materials amb unes determinades propietats.

Quan es parla de microestructura es fa referència a la disposició dels grans cristal·lins que conformen els materials i, per tant, cal emprar un microscopi que permeti observar grans de mides entre 2 i 100 micres. La mida i la disposició dels grans cristal·lins són els responsables de que un mateix material tingui propietats mecàniques diferents.

Per això, els estudiants fan diferents pràctiques per aprendre a preparar les mostres polides metal·logràficament, saber utilitzar un microscopi metal·logràfic, interpretar les imatges de les microestructures al microscopi, entendre com canvia la microestructura amb diferents tractaments tèrmics, i relacionar la microestructura amb diferents propietats mecàniques com la duresa i la resistència. A més a més, han de ser capaços de poder determinar la mida dels grans, l'índex de gra (valor de la mida promig dels grans), percentatges de fases, etc. mitjançant tractament de les imatges per ordinador, seguint les normes UNE.

1.2. Relació del TFM amb les assignatures del Màster

En relació amb el Màster TAI, impartit a la UVic, el present TFM s'insereix en les àrees d'interès de les dues assignatures següents, incidint, de ple, en la primera:

- ***Processament de Dades Multidimensionals***
- ***Processament Digital de Senyal Avançat***

Per altra banda, el procés de realització del treball es recolza en els coneixements i habilitats obtinguts en l'assignatura de ***Gestió de Projectes***, mentre que la realització de la memòria i el disseny de l'exposició final, en les assignatures ***Seminaris*** i ***Habilitats Comunicatives***.

1.3. Organització de la memòria

La memòria del TFM està organitzada en 9 capítols i un annex:

En el **Capítol 1** es fa un breu apunt dels àmbits que emmarquen el projecte.

En el **Capítol 2** s'estableixen els objectius i es fixen els requeriments que haurà de satisfer el programa.

Atès que el programa desenvolupat en aquest TFM es comporta com un instrument de mesura gràfic, caldrà conèixer l'eina que s'ha utilitzat per fer el tractament de les imatges. MATLAB disposa, com ja és sabut, d'eines matemàtiques molt potents que simplifiquen la programació a l'hora de tractar amb imatges. Tanmateix, MATLAB també disposa d'eines de programació que permeten realitzar aplicacions, i d'una eina gràfica de disseny d'interfícies d'usuari (GUIDE). En el **Capítol 3** es fa una breu introducció a l'estructura dels gràfics de MATLAB, i en el **Capítol 4** s'introdueixen les interfícies gràfiques d'usuari (GUI) i l'eina GUIDE de MATLAB.

En el **Capítol 5** s'explica el disseny de la interfície gràfica, en la que s'ha buscat sempre la facilitat d'utilització, la claredat, la funcionalitat i l'exactitud.

En el **Capítol 6** s'exposa el disseny de tots els procediments i funcions del programa. És un capítol extens, atesa la gran quantitat de funcions que s'han desenvolupat.

El **Capítol 7** és bàsic en el desenvolupament del projecte. La realització d'un instrument de mesura no té sentit si no es determina la confiança que es pot dipositar en les seves mesures. En el capítol es determina l'error i es proposen procediments de càlcul per a aquelles mesures en les que és possible fer-ho. Per a les altres, es proposen procediments alternatius o es dóna una indicació del marge de confiança que es pot esperar.

Finalment, el **Capítol 8** recull els resultats aconseguits amb el programa, s'hi proposen una sèrie de millores o ampliacions futures i es formulen les valoracions i les conclusions a les que s'ha arribat al final del treball.

Per acabar, el **Capítol 9** recull la bibliografia consultada, bàsicament pàgines web i documentació en línia.

En el **Capítol 10**, l'Annex, es llista el codi del programa PRANIMET.

2. Objectius i especificacions

El present TFM es proposa aconseguir els objectius que s'indiquen en l'apartat 2.1. Per fer-ho, l'aplicació desenvolupada haurà de satisfer els requeriments que s'estableixen en l'apartat 2.2.

2.1. Objectius

El TFM es titula ***Programa d'anàlisi d'imatges de provetes metal·logràfiques***, i el seu objectiu és explorar les possibilitats del programa matemàtic MATLAB de MathWorks i la seva eina Entorn de Disseny d'Interfície Gràfica d'Usuari (GUIDE), desenvolupant un programa d'anàlisi d'imatges de provetes metal·logràfiques que es pugui utilitzar per a realitzar les pràctiques de laboratori de l'assignatura de Tecnologia de Materials de la titulació de Grau en Enginyeria Electrònica, que s'imparteix a la Universitat de Vic.

El programa final s'haurà de compilar, de manera que pugui funcionar en qualsevol ordinador de la Universitat o de l'alumnat (amb uns mínims requeriments tècnics), sense la necessitat d'utilitzar cap altre tipus de programari comercial com a suport.

2.2. Requeriments del programa

Des del Departament d'IACA de l'Escola Politècnica Superior s'han proporcionat els requeriments que ha de satisfer el programa:

1. Es parteix d'imatges de microscòpia òptica, realitzades al Laboratori de Mecànica de la UVic mitjançant un microscopi òptic *Carl Zeiss Jena* i una càmera de fotos digital *Sony Cibershoot* adaptada al microscopi. Es disposarà de:
    - Imatges de recristal·lització del coure.
    - Imatges d'acers.
    - Imatges de foses.

2. En cada un d'aquests casos es pretén fer les següents accions i obtenir les següents mesures:

    a. Calibrar imatges de microscòpia.
    b. Mesurar empremtes de duresa i calcular la duresa dels materials.
    c. Mesurar els gruixos de les capes i altres distàncies en les mostres.
    d. Mesurar els diàmetres dels grans i calcular el seu diàmetre mig.
    e. Determinar l'índex de mida de gra.
    f. Quantificar diferents fases cristal·lines (grafit en una fosa, perlita en un acer, inclusions de cuprita en el coure, etc.)

a. Calibrar les imatges de microscòpia.

Els paràmetres més importants que defineixen les característiques d'un instrument de mesura són els que fan referència a la seva exactitud, precisió, resolució, etc. Per tant, és molt important que les imatges dels materials que es prenen pel microscopi estiguin ben calibrades.

El procés de calibrat consisteix en :

I. Fotografiar un micròmetre a través del microscopi, quan aquest últim estigui en les mateixes condicions (els mateixos augments, el mateix zoom de la càmera fotogràfica, etc.) que s'utilitzaran en les mesures de les mostres següents. És important que totes les imatges obtingudes en cada procés de mesura tinguin les mateixes resolució i dimensions.

II. Calcular el factor de calibrat de la imatge, a partir de les dimensions en píxels del micròmetre (Factor de calibrat = n µm/m píxels).

III. Aplicar el factor de calibrat calculat a cada una de les imatges corresponents a les mesures fetes en aquelles condicions.

El calibrat és el primer pas que s'ha de realitzar abans que totes les altres anàlisis, i és important que es realitzi amb la màxima rigorositat.

b. Mesurar empremtes de duresa i calcular la duresa dels materials.

A partir de les empremtes Brinell i Vickers deixades pel duròmetre *Hoytom 713 SR*, s'ha de mesurar el diàmetre de les primeres i les diagonals de les segones per calcular la duresa dels materials.

Les empremtes Brinell són circulars i cal passar obligatòriament pel centre per tenir una mesura fiable del seu diàmetre. Una vegada conegut el diàmetre de la empremta, i els paràmetres del duròmetre es pot calcular la duresa del material:

$$Duresa\ [HB] = \frac{2P}{\pi D\left[D - \sqrt{D^2 - d^2}\right]} \qquad (2.1)$$

on *D és el diàmetre de la punta del duròmetre*
d és el diàmetre de l'empremta
P és la càrrega del duròmetre

Les empremtes Vickers són romboides i cal mesurar les diagonals.

$$Duresa\ [HV] = \frac{1.854P}{d_1^2} \qquad (2.2)$$

on *d_1 és la diagonal de l'empremta*
P és la càrrega del duròmetre

Una manera fiable de fer el càlcul consisteix en ajustar una circumferència a les empremtes, i calcular el seu diàmetre.

Pel cas dels acers també caldria calcular la seva resistència (σ_u)

$$\sigma_u\ (MPa) = 3.5\ HB \qquad (2.3)$$

c. Mesurar els gruixos de les capes i altres distàncies en les mostres.

Cal dissenyar una eina que permeti obtenir una mesura fidel de les distàncies, longituds i diàmetres de grans, gruixos de capes i altres ocurrències de les mostres.

Tant la línia de cota que identifiqui la mesura, com el seu valor, hauran de quedar marcats a la imatge per a posteriors impressions o còpies.

d. Mesurar els diàmetres dels grans i calcular el seu diàmetre mig.

Cal determinar el contorn de cada gra de la imatge i processar la informació per a trobar el diàmetre mig dels grans.

També cal determinar la mida de gra aparent dels acers ferrítics o austenítics segons la normativa UNE-En ISO 643. Aquesta normativa descriu els mètodes per destacar o revelar les vores dels grans i per estimar el diàmetre mig dels grans d'un acer amb distribució unimodal.

La mida de gra aparent, segons aquesta normativa, es pot determinar per:

a. Per l'índex de mida del gra,
   ja sigui per comparació amb imatges tipus per la mesura de la mida dels grans,
   ja sigui per comptatge de grans amb la finalitat de determinar el número promig de grans per unitat d'àrea.
b. Pel valor promig dels segments o dels grans interceptats.

e. Determinar l'índex de mida de gra.

Determinar aquest índex a partir de les instruccions donades en la normativa UNE-En ISO 643.

f. Quantificar diferents fases cristal·lines (grafit en una fosa, perlita en un acer, inclusions de cuprita en el coure, etc.)

Per quantificar les diferents fases cristal·lines cal que tinguin un aspecte diferent, de manera que la imatge es pugui binaritzar. En el cas de les foses (o fundicions) és molt senzill, i un procés manual per aconseguir-ho pot ser el següent:

a. Transformar la imatge a escala de grisos.
b. Ajustar el contrast i la brillantor.
c. Binaritzar la imatge, ajustant el llindar de manera que la imatge resultant sigui el més semblant possible a l'original.
d. Quantificar els píxels de cada fase, i obtenir el percentatge.

3. Estructura dels gràfics de MATLAB

Els ***Objectes Gràfics*** són els elements bàsics de dibuix utilitzats per MATLAB per a visualitzar les dades. Cada instància d'un objecte s'associa a un únic identificador anomenat ***handle***. Utilitzant aquest *handle*, es poden manipular les característiques, anomenades ***propietats dels objectes*** (*object properties*) de qualsevol objecte gràfic existent. En el moment de la creació d'un objecte ja es poden especificar els valors de les seves propietats.

Aquests objectes s'organitzen de manera jeràrquica, com es mostra a la figura següent:

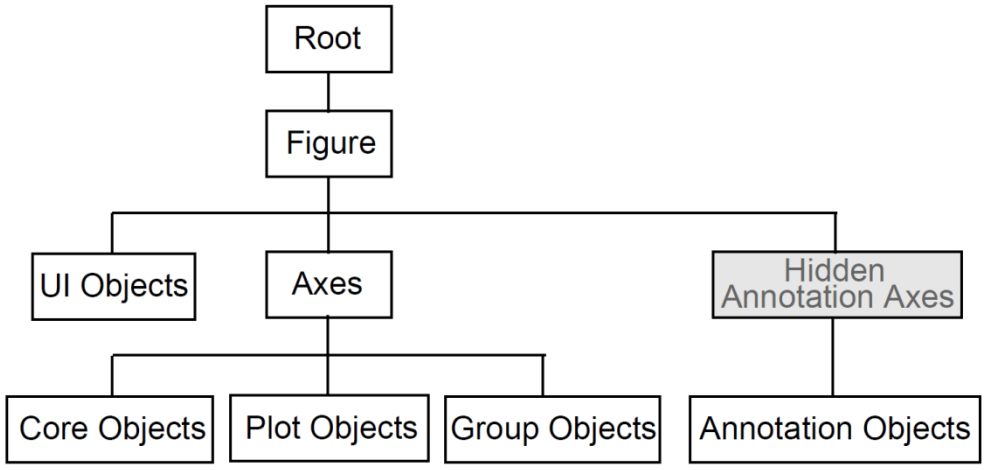


Figura 3.1. *Organització jeràrquica dels Objectes gràfics de MATLAB.*

La naturalesa jeràrquica dels ***Handle Graphics*** es basa en les interdependències dels diferents tipus d'objectes gràfics. Per exemple, per dibuixar una Línia (***object Line)***, MATLAB necessita un Eix (***object Axes)*** per orientar i proveir un marc de referència a la línia. Els Eixos, al seu torn, necessiten una finestra de referència, anomenada Figura (***object Figure)***, per a posicionar i dibuixar els Eixos i tots els seus objectes fills.

3.1. Tipus d'objectes gràfics

Hi ha dos tipus d'objectes gràfics:

- **Objectes gràfics bàsics**, són els objectes gràfics elementals, i s'utilitzen per les funcions de dibuix d'alt nivell, i pels objectes compostos per crear objectes de dibuix.
- **Objectes compostos**, estan compostos d'objectes gràfics bàsics que s'han agrupat per oferir una interfície més convenient a l'usuari.

Els objectes compostos constitueixen la base de quatre subcategories d'objectes gràfics:

- **Objectes de dibuix** *(plot objects)*, compostos per objectes gràfics bàsics, que s'agrupen de manera que les propietats de tots ells es fixaran, en conjunt, quan es fixin les propietats de *l'objecte de dibuix*.
- **Objectes d'anotació** *(annotation objects)*, que es situen en una capa separada dels altres objectes gràfics.
- **Objectes grup** *(group objects)*, que són objectes formats per un grup d'objectes que es comporten, en determinats aspectes, com si fossin un sol objecte.
- **Objectes UI** *(UI objects)*, que són els objectes que s'utilitzen per a construir les interfícies gràfiques d'usuari.

Tots aquests objectes gràfics són interdependents, de manera que una pantalla gràfica normalment conté una certa varietat d'objectes que, en conjunt, produeixen un gràfic o una imatge significativa.

3.2. L'objecte Arrel *(Root Object)*

L'*Arrel* és el pare de totes les *Figures* i de tots els altres objectes. No es pot instanciar, atès que només serveix per guardar informació de l'estat de MATLAB i del sistema. Només hi ha una sola *Arrel*, i no cal declarar-la doncs es crea en el moment d'iniciar MATLAB.

De totes maneres, l'usuari pot modificar les seves propietats, modificant així les propietats de totes les *Figures* filles.

El *handle* de l'arrel sempre pren el valor 0.

3.3. La finestra gràfica: la *Figura*

Les *Figures* són les finestres gràfiques que utilitza MATLAB per mostrar els gràfics i les dades. Les *Figures* poden contenir menús, barres d'eines, objectes d'interfície amb l'usuari, menús contextuals, eixos i, com a fills dels eixos, tots els altres tipus d'objectes gràfics. No hi ha límits en quant a la quantitat de *Figures* que es poden crear.

Les *Figures* juguen dos tipus de rols diferents en MATLAB:

- Contenir gràfics de dades
- Contenir interfícies gràfiques d'usuari

En realitat, les *Figures* poden contenir ambdós components al mateix temps. La figura següent il·lustra els tipus d'objectes que pot contenir una *Figura*:

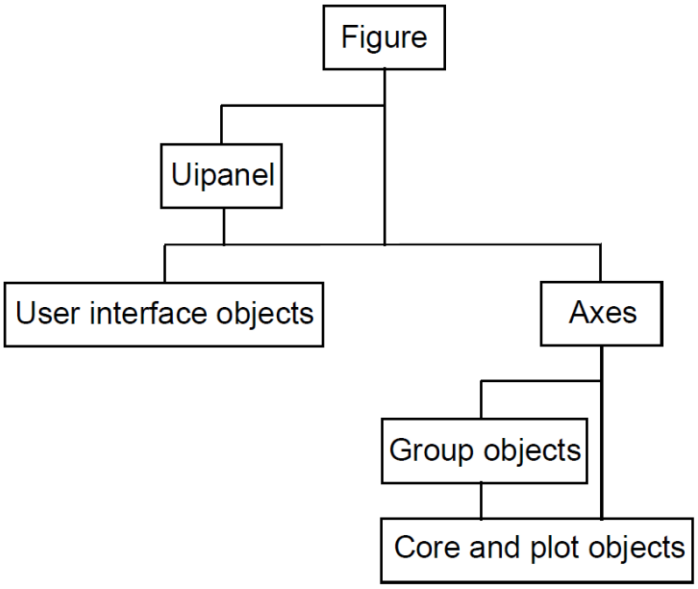


***Figura 3.2**. Tipus d'objectes que pot contenir una Figura de MATLAB.*

Es pot observar que tant les *Figures* com els *Axes* tenen fills que actuen com a contenidors d'altres objectes. Un ***UIpanel*** (panel d'interfície d'usuari) conté objectes d'interfície amb l'usuari, i està emparentat (és fill) d'una *Figura*. Per altra banda, els objectes grup poden contenir *Eixos* fills, i ser fills d'un eix superior.

Els *handles* de les *Figures* són nombres enters (1, 2, 3...) o de coma flotant, i el comandament GCF (*Get Current Figure*) pren el valor del *handle* de la *Figura* que està activa en aquell moment.

Els *fills* de les *Figures* per als gràfics

Les *Figures* que mostren gràfics necessiten contenir *Eixos* per proveir de finestres de referència a objectes tals com L*ines* i S*urfaces*, que s'utilitzen per a representar dades. Aquestes dades poden estar contingudes en objectes grup, o directament en els *Eixos*.

Les *Figures* poden contenir múltiples *Eixos* situats en qualsevol posició de la *Figura,* i de qualsevol dimensió (que estigui continguda en les dimensions de la *Figura*, lògicament)

Les *Figures*, utilitzades per les Interfícies Gràfiques d'Usuari (*GUI*)

Les GUIs poden ser des de simples caixes de diàleg d'atenció, fins a sofisticades aplicacions. Es poden modificar molts aspectes d'una *Figura* per adaptar-la als usos previstos modificant les seves propietats. Per exemple, les següents propietats de les *Figures* sovint són útils quan es creen interfícies d'usuari:

- Mostrar o amagar els menús de la *Figura*, mentre es mostren els menús creats per l'usuari (*MenuBar*).
- Modificar el títol de la *Figura* (*Name*).
- Controlar l'accés de l'usuari al *handle* de la *Figura* (*HandleVisibility*).
- Crear una subrutina que s'executi quan l'usuari redimensioni la *Figura* (*ResizeFcn*).
- Controlar la visualització de la barra d'eines de la *Figura* (*Toolbar*).
- Assignar un menú contextual (*UIContextMenu*).
- Definir *callbacks* (crides a procediments) que s'executin quan els usuaris cliquin, arrosseguin o alliberin el ratolí damunt la *Figura* (*WindowButtonDownFcn, WindowButtonMotionFcn, WindowButtonUpFcn*).
- Especificar quan la *Figura* és modal (*WindowStyle*).

3.4. Els objectes gràfics bàsics (*Core Graphics Objects*)

Els objectes gràfics bàsics inclouen primitives bàsiques de dibuix com *line*, *text* i polígons tancats (*patch objects*). També inclouen altres elements més complexes com *surfaces*, *images* i objectes de llum.

Els *Eixos* contenen objectes que representen dades, com *line*, *surface*, *contourgroups*, etc.

El *handle* dels objectes gràfics bàsics sempre pren valors de coma flotant (per exemple, 1.24589).

La ***Taula 3.1*** llista i defineix els objectes gràfics bàsics:

Taula 3.1. *Propòsit dels objectes gràfics bàsics en MATLAB*

Funció	Propòsit
Axes	Els objectes *Axes* defineixen les coordenades per a visualitzar gràfics. Sempre estan continguts en una *Figura*
Image	Són representacions 2-D d'una matriu en la que els valors numèrics es mapen amb colors. Les *Image* també poden ser matrius 3-D de valors RGB.
Light	És una font de llum direccional localitzada en l'interior del *Axes*. Les llums afecten els plegats i les superfícies, però no poden ser vistes.
Line	Les línies es dibuixen connectant les dues dades o punts que les defineixen.
Patch	Són polígons plens, amb les propietats de les vores diferenciades. Un *Patch* senzill pot contenir múltiples cares, cada una d'elles acolorida amb colors sòlids o extrapolats.
Rectangle	És un objecte 2-D en el que es poden configurar les vores, els colors i la curvatura (es poden dibuixar el·lipses).
Surface	Graella o quadrilàter 3-D creat per dibuixar el valor de cada element d'una matriu com a alçada sobre el plànol x-y
Text	Cadena de caràcters posicionats en el sistema de coordenades definit per l'*Axes*.

La figura següent il·lustra alguns objectes gràfics bàsics:

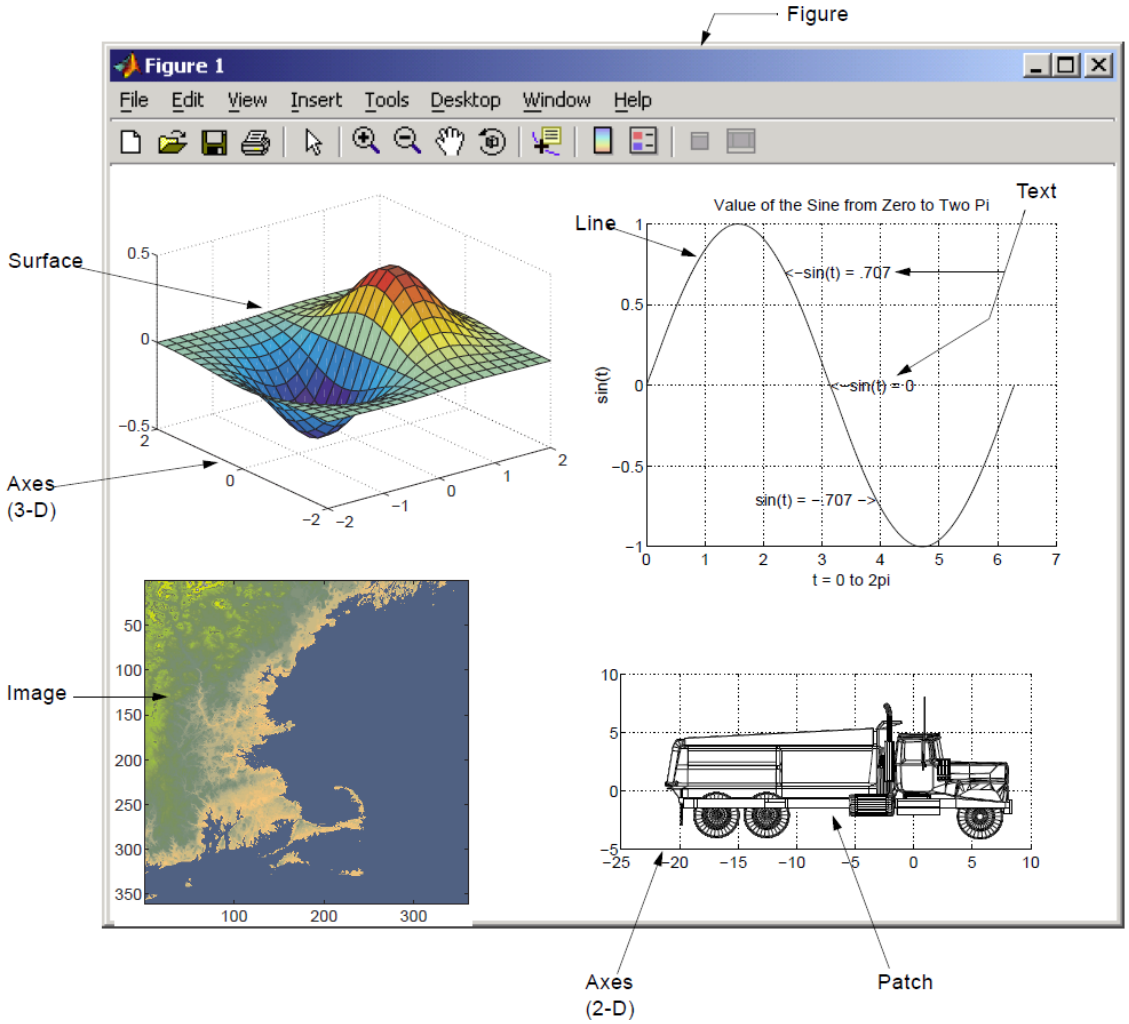


Figura 3.3. *Exemple d'alguns objectes gràfics bàsics de MATLAB.*

La funció que s'utilitza per crear un objecte gràfic té la forma següent:

handle = funció('nom_de_la_propietat' , 'valor_de_la_propietat' , ...)

Es poden especificar els valors de les diferents propietats de l'objecte (sempre per parelles -nom i valor de la propietat-), i la funció retorna el valor del *handle* que identifica l'objecte que s'ha creat. Amb aquest *handle* es poden demanar i/o modificar les propietats de l'objecte un cop ja s'ha creat.

El següent exemple avalua una determinada funció matemàtica, i crea tres objectes gràfics utilitzant els valors de les propietats especificats com a arguments de les instruccions *Figure*, *Axes* i *Surface*. MATLAB fa servir els valors que té per defecte per a totes les altres propietats que no estan especificades.

```
1.  [x,y] = meshgrid([−2:.4:2]);
2.  Z = x.*exp(−x.^2−y.^2);
3.  fh = figure('Position',[350 275 400 300],'Color','w');
4.  ah = axes('Color',[.8 .8 .8],'XTick',[−2 −1 0 1 2],'YTick',[−2 −1 0 1 2]);
5.  sh = surface('XData',x,'YData',y,'ZData',Z,...
        'FaceColor',get(ah,'Color')+.1,...
        'EdgeColor','k','Marker','o',...
        'MarkerFaceColor',[.5 1 .85]);
```

La línia 3 crea una *Figura* que es posicionarà en la pantalla en posició (350 , 275), tindrà una amplada/alçada de 400/300, i el color de fons serà blanc (w). El seu *handle* el desa a fh.

Si en un moment determinat es volgués canviar el color de fons de la *Figura* a vermell, s'indicaria:

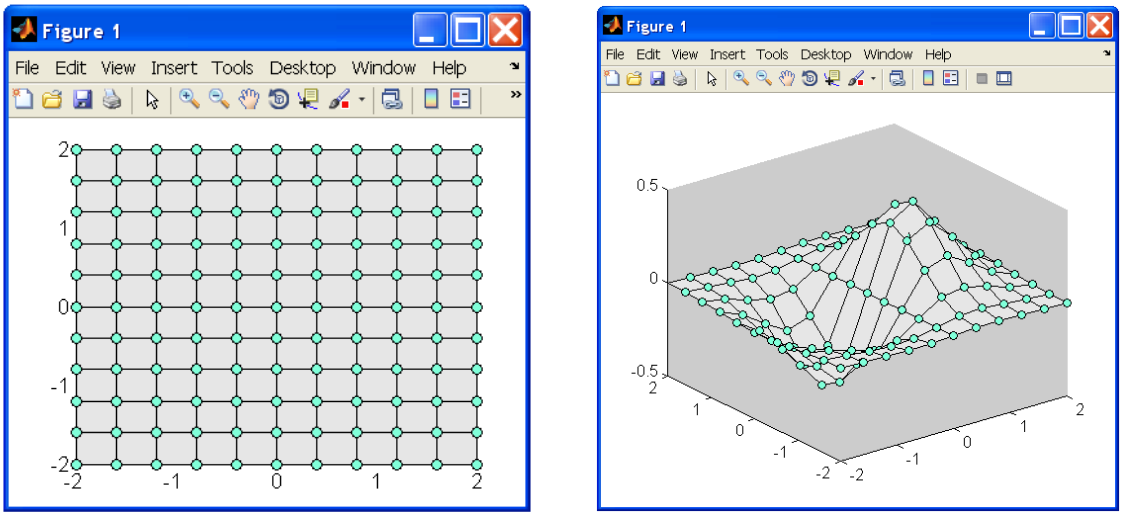
```
set (fh , 'Color' , 'r');
```

La línia 4 crea uns *Eixos* x-y amb marques a les posicions (-2, -1, 0, 1, 2), i amb un color de fons que es correspon a 0.8R, 0.8G i 0.8B. El seu *handle* el desa a ah.

La línia 5 dibuixa una *Surface* amb la matriu generada en la línia 1 (es correspon a una graella quadrada, entre -2 i +2, amb una separació entre punts de 0.4 en direcció x i en direcció y). Fixa el color de fons de la graella (*FaceColor*) com el color de fons de l'*Axes* (get(ah , 'Color')) més 0.1. També fixa el color de les línies (*EdgeColor*) de color negre (*blacK*), la forma dels nodes de la malla com una 'o' i el color d'aquests nodes (*MarkerFaceColor*) del color R=0.5, G=1 B=0.85. El *handle* d'aquesta superfície el desa a sh.

En la ***Figura 3.4.a*** es pot veure el resultat d'executar les instruccions anteriors.

Es pot observar que la funció *surface* no dibuixa una representació 3-D, sinó que només dibuixa una representació x-y. Si es vol veure l'efecte de la funció de la línia 2 aplicada sobre la coordenada z s'ha de variar el punt de vista de l'*Axes* amb la sentència view(3). En la ***Figura 3.4.b*** es pot veure el seu resultat.



a) b)

Figura 3.4. *Resultat de l'execució de les instruccions anteriors: a) vista 2-D. b) vista 3-D.*

Per defecte, totes les sentències que creen objectes gràfics ho fan en la *Figura* corrent (o actual) i en l'*Eix* corrent (si l'objecte que es crea és fill d'un *Eix*). Tanmateix, a l'hora de crear un objecte es pot especificar el seu pare, i no cal que sigui el corrent. Per exemple, la sentència

```
axes('Parent' , figure_handle , ...)
```

crea un *Eix* en la *Figura* identificada pel *handle* '*figure_handle*'.

També es pot moure un objecte des d'un 'Pare' a un altre, redefinint la seva propietat 'Parent':

```
set(gca , 'Parent' , figure_handle)
```

Aquesta sentència mou l'*Eix* actual (gca) al 'Pare' *figure_handle.*

4. Les GUIs i l'eina GUIDE de MATLAB

Quan es desenvolupa un programa o una aplicació, sigui quina sigui l'eina amb la qual es desenvolupi, sovint fa falta que els usuaris interactuïn amb aquests programes, ja sigui per entrar dades, opcions de configuració o ordres de funcionament, ja sigui per llegir, de manera adequada la informació elaborada pels programes. El codi del programa (i tota la seva aparença i funcionament) que permet que un usuari interactuï amb aquest programa constitueix la interfície d'usuari del programa (IU).

Antigament, les interfícies d'usuari eren molt simples, i consistien en visualitzar les dades, en forma de text, línia a línia, per la pantalla, i entrar les dades i les ordres pel teclat.

Amb l'increment de prestacions dels ordinadors s'han creat interfícies d'usuari que utilitzen els recursos gràfics dels propis ordinadors de manera intensiva i que resulten molt més amigables que els anteriors. També proporcionen més informació, i de manera més clara. El propi sistema operatiu Windows conté una GUI molt elaborada (copiada en una bona part del sistema operatiu d'Apple), que permet que l'usuari interactuï amb el Windows de manera fàcil, còmoda i segura (de vegades...).

Així doncs, les GUIs són tot el conjunt de recursos gràfics i de programació del propi programa que s'està executant que permeten que l'usuari interactuï amb ell, tal com hem dit, de manera fàcil, còmoda i segura.

Avui dia, pràcticament tots els entorns de programació disposen d'eines per a generar interfícies gràfiques d'usuari. MATLAB també en disposa.

4.1. La Interfície Gràfica d'Usuari de MATLAB

En el capítol anterior s'ha vist que una *Figura* pot contenir dos tipus d'objectes:

- Objectes gràfics, tals com *Axes, line, rectangle, text*, etc.
- Controls de la Interfície Gràfica d'Usuari (GUIs).

La figura següent, que és una fotografia d'una pantalla del TFM que es presenta en aquesta memòria, mostra clarament els dos tipus d'objectes:

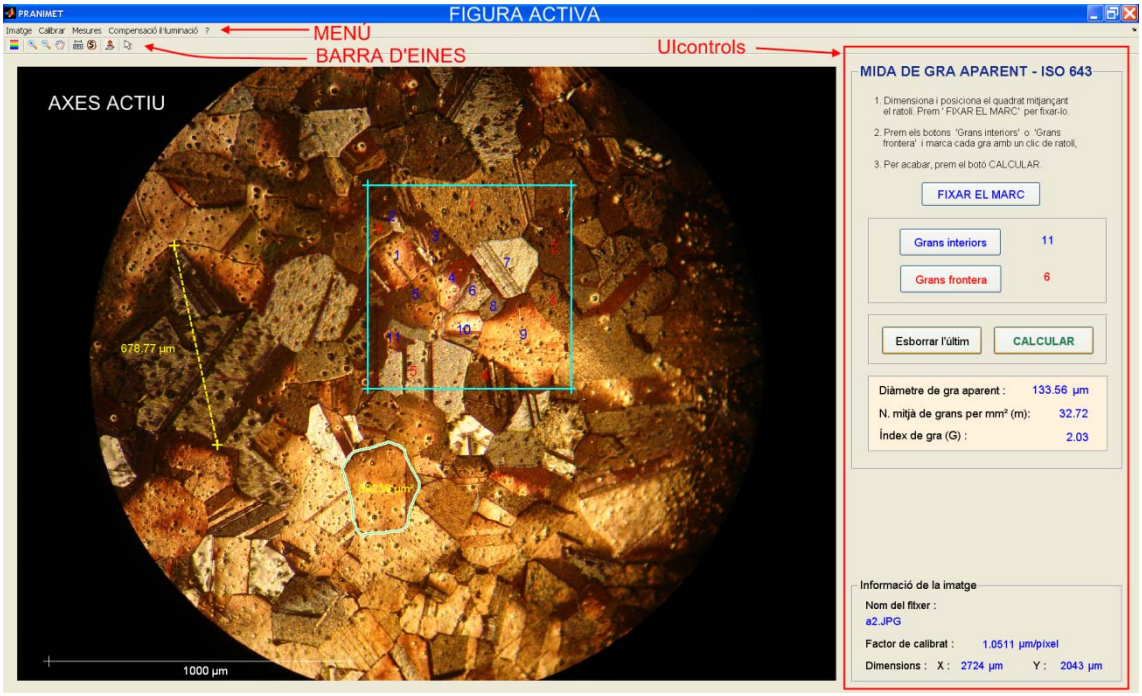


Figura 4.1. *Interfície Gràfica d'Usuari del present TFM.*

- El programa té una sola *Figura*, que es correspon amb la imatge que s'observa per la pantalla. Evidentment, es tracta de la *Figura* activa.
- La *Figura* conté tres grups diferents d'objectes:

    1. Per una banda conté un sol *Eix*, que ocupa una bona part de la pantalla, i que conté tots els objectes gràfics que permet mostrar el programa (*imatges, línees, rectangles, text*...).

    2. En la part superior de la pantalla es troba una barra de menús i una barra d'eines diverses.

    3. La part dreta de la pantalla conté una sèrie de quadres de text, botons per prémer, etc, que controlen el funcionament del programa. Són els *UIcontrols,* o objectes gràfics de control.

En aquesta ***Figura 4.1*** s'observa clarament l'estil del GUI de MATLAB, molt semblant al que proporciona el propi Windows o els llenguatges actuals de programació. El GUI és tot el que es veu, amb la circumstància que un dels elements del GUI són uns *Eixos* que contenen una imatge i una colla d'elements gràfics. Tot el demés que apareix per la pantalla són elements de control del GUI (*menús, toolbars, pushbuttons, quadres de text*, etc.).

Es pot veure com alguns d'aquests objectes estan situats dintre d'un rectangle (per exemple, els botons '**Grans interiors**' i '**Grans exteriors**', o bé tot el text etiquetat com '**Informació de la imatge'**). Aquests rectangles són uns altres dels objectes del GUI, anomenats *Panels,* la missió dels quals és contenir altres objectes del GUI dotant-los d'unes propietats comunes. En aquest cas, si es desactiva el panel corresponent a la **'Informació de la imatge'**, desapareix de la vista tota la informació continguda en el seu interior.

Objectes gràfics del GUI de MATLAB

Els objectes gràfics del GUI de MATLAB poden ser:

1. *Eixos* que contenen imatges i gràfics.
2. Menús contextuals i Barres d'eines configurables per l'usuari.
3. Controls del GUI o ***UIcontrol.***

Text Estàtic. Pot mostrar símbols, missatges i valors numèrics, i es pot situar en qualsevol lloc de la *Figura*. El *Text Estàtic* no té cadenes d'invocació.

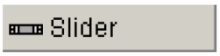
Text Editable. Aquest dispositiu permet a l'usuari teclejar una cadena d'entrada.

Menú Desplegable. Poden aparèixer en qualsevol situació de la *Figura*, i permeten escollir una opció entre totes les que es despleguen quan es prem el botó.

Push Button. Els *Polsadors* generen una acció quan es clica damunt seu amb el ratolí, o quan s'allibera el clic del ratolí.

Casella de verificació. Estan dissenyades per a realitzar operacions d'encesa/apagada. Quan la casella està activada queda marcada amb un senyal. Quan no ho està, la marca desapareix.

Botó de radi. Són similars a les *Caselles de verificació*, però quan s'utilitzen en grup són mútuament excloents (si un botó està activat, tots els demés estan desactivats).



Slider o barra cursor. Accepten dades d'entrada numèriques dins d'un rang específic. Els usuaris mouen la barra pressionant el botó central i arrossegant-lo. Al final, la posició del cursor en la barra donarà un determinat valor numèric.

Panel. Serveixen per agrupar altres objectes i conferir-los propietats comunes.

Botó de palanca. Generen accions que indiquen un estat binari. Quan es premen per primera vegada apareixen com a 'on'. Quan es premen per segona vegada apareixen com a 'off'.

Caixa de llista, o llista. Mostra una sèrie d'articles i permet a l'usuari seleccionar-ne un o més d'un.

Les crides a funcions dels objectes del GUI

La manera de treballar de les aplicacions generades amb MATLAB és semblant a les aplicacions generades amb programació orientada a objectes. El programa consta d'una sèrie de procediments o funcions que només s'activaran en el moment que es produeixi un determinat EVENT.

MATLAB anomena *Callbacks* a les crides a funcions en el moment que es produeix un event. Existeixen diferents tipus de *Callbacks*.

1. **Callbacks de tots els objectes gràfics**. Tots els objectes gràfics tenen tres propietats per a les quals es poden definir rutines que es cridaran en el moment que es produeixi l'event:

    - ***ButtonDownFcn***. Es cridarà la funció en el moment que es cliqui el botó del ratolí quan el cursor estigui damunt de l'objecte gràfic.
    - ***CreateFcn***. Es cridarà la funció en el moment que es creï l'objecte, i abans que s'executi cap altre event.
    - ***DeleteFcn***. Es cridarà la funció en el moment que s'esborri l'objecte.

2. ***Callbacks* dels objectes *UIcontrol* i *UImenu*.** A més a més dels *Callbacks* anteriors, tots els objectes *UIcontol* i *UImenu* tenen altres propietats que generaran *Callbacks*, en funció de la seva naturalesa. Per exemple, un *PushButton* tindrà els següents:

    - ***ButtonUpFcn***. Cridarà a la funció en el moment que es deixi de clicar el botó del ratolí quan el cursor estigui damunt de l'objecte gràfic.
    - ***ButtonMotionFcn***. Cridarà a la funció en el moment que el cursor del ratolí passi pel damunt de l'objecte, sense clicar.

3. ***Callbacks* de les *Figures*.** Les *Figures* tenen propietats extres que generaran *Callbacks*:

    - ***CloseRequestFcn***. Cridarà a la funció en el moment que es tanqui la *Figura.*
    - ***KeyPressFcn***. Cridarà a la funció en el moment que l'usuari premi una tecla del teclat.

- ***ResizeFcn***. Cridarà a la funció en el moment que l'usuari variï les dimensions de la finestra gràfica.
- ***WindowButtonDownFcn***. Cridarà a la funció en el moment que es cliqui el ratolí quan el ratolí estigui damunt de la *Figura*.
- ***WindowButtonUpFcn***. Cridarà a la funció en el moment que es deixi de clicar el ratolí quan estigui damunt de la *Figura*.
- ***WindowButtonMotionFcn***. Cridarà a la funció en el moment que el cursor del ratolí es mogui pel damunt de la *Figura,* sense clicar.

En l'exemple de sota es poden veure tres instruccions.

La primera crea un objecte *UIcontrol*, exactament un *Pushbutton*, situat en la posició [30,110,120,30] de la *Figura* actual i que portarà el text 'Engegar' escrit al damunt. El *handle* del *Pushbutton* es desa a la variable hp1.

En el moment que s'engegui el programa i es creï el *Pushbutton*, el programa farà una crida a la funció **engegar_CreateFcn** (segona instrucció de l'exemple) i l'executarà. A partir d'aquest moment no es tornarà a cridar cap més vegada aquesta funció.

Durant el funcionament normal del programa, cada vegada que es premi el botó del ratolí quan el cursor estigui damunt del *Pushbutton* es cridarà la funció **engegar_Callback** (tercera instrucció de l'exemple) i l'executarà.

```
hp1 = uicontrol(gcf , 'Style' , 'pushbutton' , 'Position' , [30,110,120,30],...
                    'String' , 'Engegar' , 'Callback' , 'engegar');

function engegar_CreateFcn(hObject, eventdata, handles)
% Instruccions que s'executaran en el moment de crear el Pushbutton,
% i abans d'altres crides

Function engegar_Callback(hObject, eventdata, handles)
% Instruccions que s'executaran cada vegada que es cliqui el botó
% del ratolí damunt del Pushbutton
```

4.2. L'eina GUIDE

GUIDE (*Graphical User Interface Development Environment*) és un joc d'eines de MATLAB dissenyades per a crear GUIs (*Grafical User Interfaces*) de manera fàcil i ràpida. Funciona de manera similar a entorns de programació com VisualBasic, VisualC, etc.

Una aplicació GUIDE genera dos arxius: *.m* i *.fig*. L'arxiu *.m* conté el codi de totes les accions que farà el programa quan s'accionin els botons de control de la interfície, mentre que el fitxer *.fig* conté tots els elements gràfics, amb les seves propietats inicials. Els dos fitxers (*.fig* i *.m*) han d'anar sempre junts.

La creació d'una aplicació mitjançant GUIDE segueix els passos següents:

1. S'engega l'eina GUIDE i es dóna un nom al projecte. El sistema hi afegeix l'extensió *.fig*.
2. Es fixen les dimensions de la pantalla de treball, arrossegant-la amb el ratolí.
3. Es van dipositant els diferents *UIcontrols* (botons, cursors, quadres de text,...) sobre la pantalla o figura de treball mitjançant el ratolí, i es dimensionen arrossegant les vores. També es dipositen els *Eixos* que convingui de la mateixa manera.
4. Fent doble clic damunt dels objectes s'accedeix a una pàgina en la que es poden editar les propietats de cada objecte.
5. Una vegada que els controls estan en la seva posició, s'editen les funcions de crida de cada un d'ells. Clicant amb el botó de la dreta sobre un objecte s'obre una finestra en la que apareixen totes les crides a funcions que permet aquell objecte. Si es selecciona una d'aquestes *Callbacks*, aleshores el mateix programa edita un fitxer *.m* amb el mateix nom que el fitxer *.fig* i insereix la capçalera de la funció cridada.
6. A continuació, i treballant ja des de l'editor de MATLAB, s'introduiran les instruccions pertinents a cada funció creada amb GUIDE.
7. De la mateixa manera que es creen *Objectes de control* i *Eixos*, també es poden crear *Barres de menús* i *Barres d'eines*.

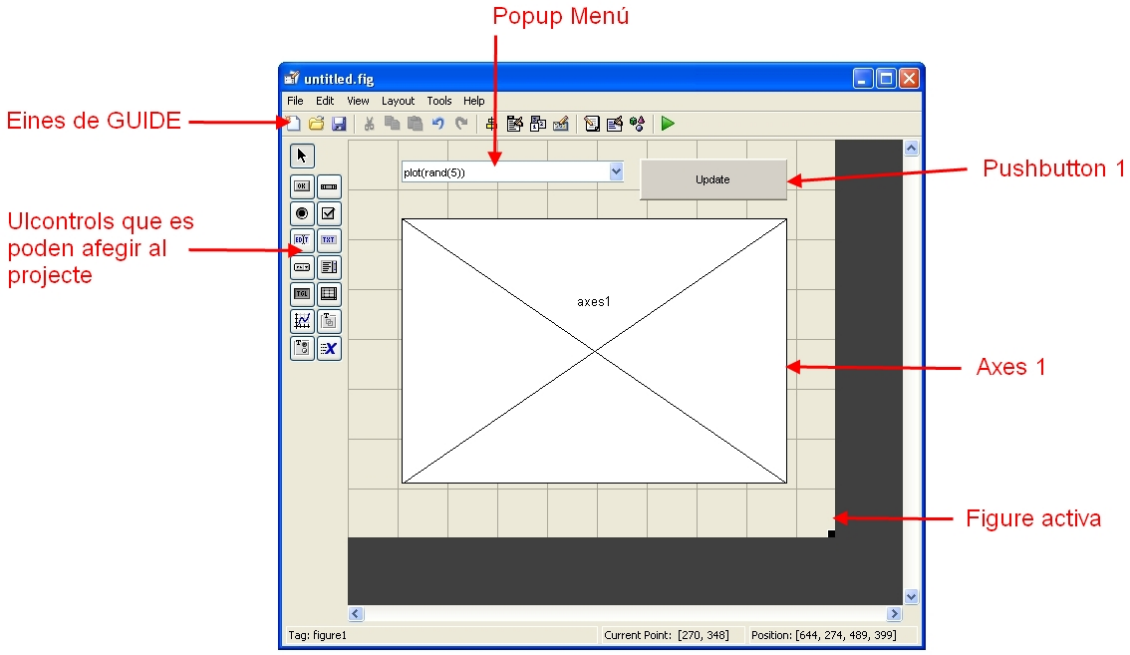


***Figura 7**. L'eina GUIDE de MATLAB.*

Avantatges i inconvenients de programar amb GUIDE

Els avantatges que suposa treballar amb GUIDE respecte a programar manualment tots els objectes gràfics són:

1. El disseny de la GUI és molt més fàcil i visual. És molt més senzill arrossegar un objecte fins a la posició desitjada i donar-li les dimensions volgudes mitjançant dos clics del ratolí, que no pas escriure la seva funció, aventurar la posició que tindrà sobre la figura i les seves dimensions, visualitzar el resultat executant el programa, i tornar enrere les vegades que calgui per ajustar la seva posició i dimensions.

2. GUIDE té un editor de propietats dels objectes molt útil. Permet modificar totes les propietats dels objectes 'in situ', sense haver-ne de recordar la sintaxi ni el seu significat.

3. La interacció entre GUIDE i el fitxer *.m* fa que aquest últim fitxer s'actualitzi cada vegada que es desa o s'executa el fitxer *.fig*. Això vol dir que quan es creen o es modifiquen objectes o es fan noves crides als objectes mitjançant GUIDE, automàticament s'afegeixen al fitxer *.m*. Tanmateix, no passa el mateix a l'inrevés, o quan s'esborren objectes o *Callbacks* en cap dels dos sentits.

Malauradament, GUIDE també té algun inconvenient com, per exemple, que tota la programació efectuada per crear el GUI i els seus objectes queda amagada als ulls del programador, en el fitxer *.fig*, sense la possibilitat de modificar-la manualment.

Això ha provocat, per exemple, que tres vegades durant el procés de creació del present programa algun semàfor del fitxer *.fig* ha quedat mal tancat, i no hi ha hagut manera de recuperar la informació introduïda des de l'última còpia de seguretat.

GUIDE també presenta problemes a l'hora de situar els *Panels* (que són contenidors d'*Objectes de control*): si el punt d'arrossegament d'un *Panel* (el punt del *Panel* per on el ratolí l'agafa i el mou) es deixa en l'interior d'un altre *Panel*, queda englobat dintre d'aquest segon. Això provoca que si en una aplicació s'han de col·locar varis *Panels* en una mateixa zona de la pantalla, i es vol que siguin independents, s'ha d'estudiar molt bé l'ordre de col·locació i les seves dimensions.

5. Disseny de la interfície gràfica

El primer pas que cal fer a l'hora de desenvolupar el programa és el disseny general de la interfície gràfica.

Els objectius i restriccions que s'han fixat per a aquest disseny són els següents.

1. La interfície amb l'usuari ha de ser molt clara, senzilla i neta. No ha de contenir elements d'ornamentació, a banda dels imprescindibles per a agrupar els comandaments.

2. Atès que l'objectiu bàsic del programa és mesurar partícules i altres elements que apareixen en una imatge, caldrà que la imatge que es mesuri sigui el més gran possible, deixant només l'espai just per a situar-hi els comandaments i els panels amb els resultats de les mesures. Així mateix, la imatge es presentarà amb les seves proporcions originals, sense cap distorsió.

3. No es pretén pas fer un programa genèric que serveixi per a moltes coses, sinó unes quantes rutines que mesurin propietats molt concretes de les imatges. Així doncs, amb l'objectiu de simplificar la seva utilització i deixar el màxim espai possible per a la imatge, el programa generarà una pantalla diferent per a cada una de les mesures que faci. D'aquesta manera, a cada pantalla només hi haurà els controls imprescindibles per a executar les corresponents mesures.

4. Atès que es pretén utilitzar el programa en la docència, es procurarà que els comandaments siguin clars en la seva funció, i que la seva distribució espacial suggereixi els passos que s'han de fer per a efectuar la mesura correctament. Per exemple, si un procés de mesura consta de tres passos que s'han de fer de manera correlativa, els botons de comandament corresponents estaran situats un sota de l'altre, i en l'ordre adequat.

5. Hi haurà dues barres superiors. La primera serà la barra de menús que permetrà, per una banda, efectuar les accions típiques amb els fitxers (obrir, tancar, desar, imprimir...) i per l'altra permetrà engegar cada un dels processos de mesura que permeti el programa. En la segona barra, la de les eines, hi haurà totes aquelles eines que funcionin indistintament amb tots els processos de mesura (eines de zoom, mesura de longituds, etc)

Després d'avaluar diferents distribucions, se n'ha escollit una com la més apropiada pel programa.

La ***Figura 5.1.*** mostra el disseny de la interfície d'usuari, amb els tres blocs operatius:

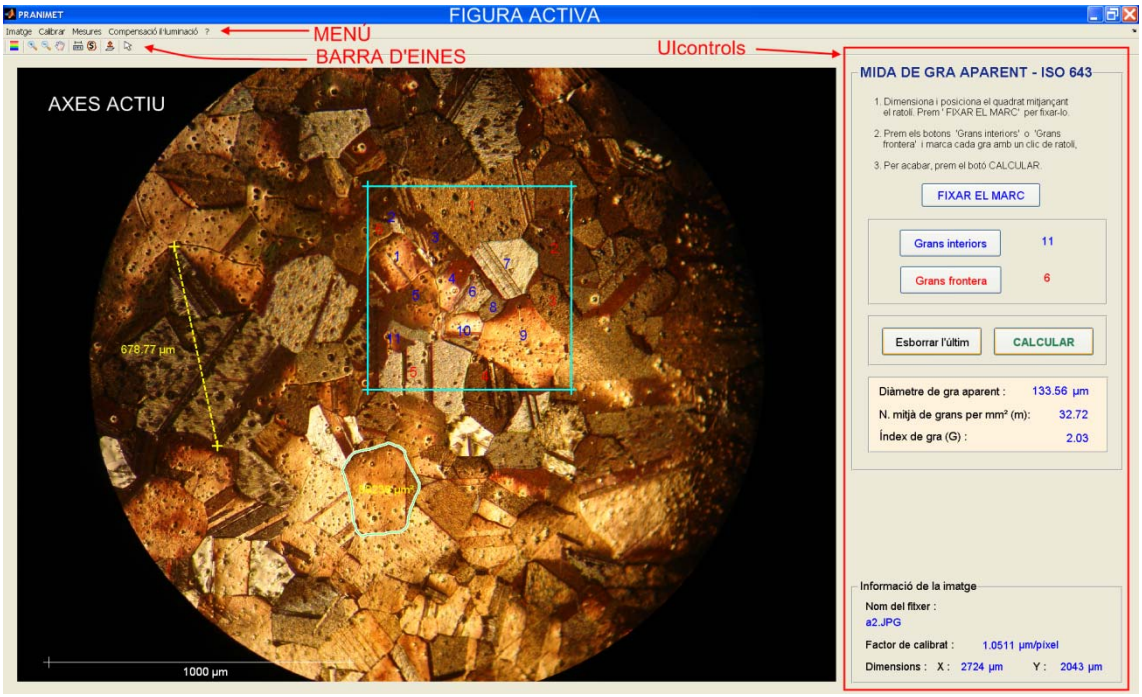


Figura 5.1*. Disseny de la interfície d'usuari.*

Bloc 1, els menús: Ocupa les tres línies superiors de la pantalla.

Aquesta posició és obligada, atès que GUIDE no deixa posar les barres enlloc més. La primera línia conté el nom del programa i els 3 botons Windows de minimitzar, maximitzar i tancar la finestra. A l'esquerra de la línia es pot veure l'anagrama del programa MATLAB. La segona línia és la barra de menús, i la tercera línia és la barra d'eines. Aquest bloc és fix i no es modifica al llarg del programa.

Bloc 2, la imatge: Ocupa la major part de la superfície de la pantalla, i està situat a l'esquerra.

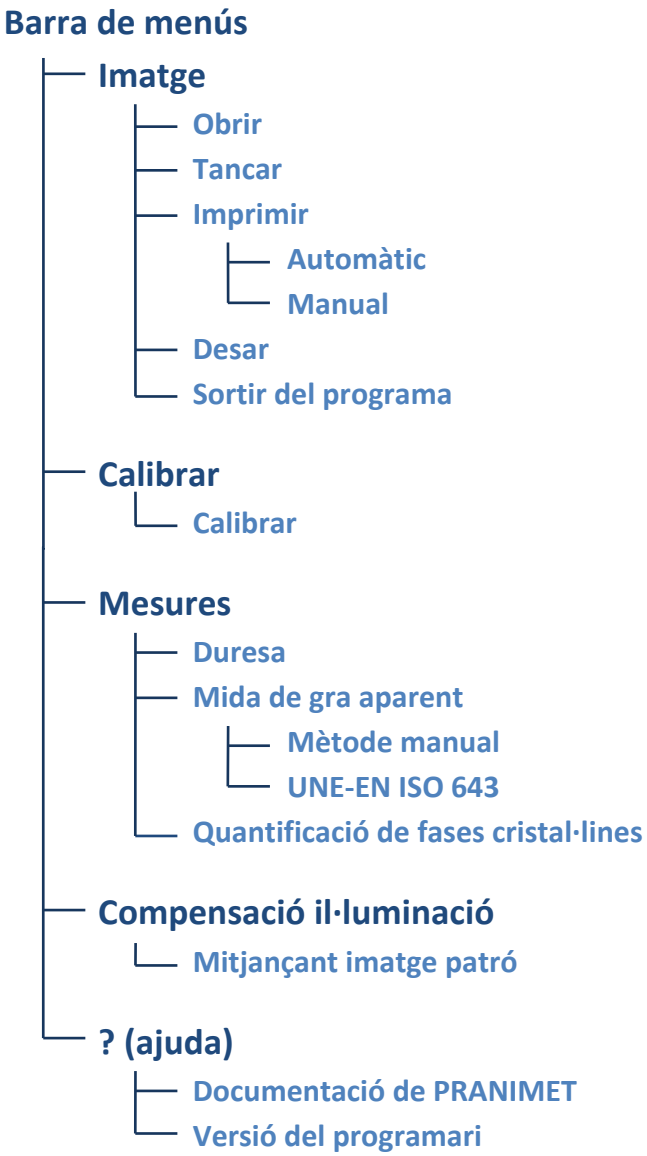
Aquest bloc és fix, i sempre conté la imatge que s'està mesurant.

Bloc 3, els comandaments: Ocupa la dreta de la pantalla

Conté els botons de comandament i els quadres de text d'entrada i visualització de dades. Aquest bloc canviarà en funció de la mesura que es realitzi en cada moment.

5.1. Disseny de la barra de menús

Atesos els requeriments inicials del programa, els menús s'han dissenyat de manera que agrupin les diferents accions i mesures que es poden fer amb les imatges.



Tal com es demanava en els requeriments del disseny de la interfície, es pot observar que la distribució dels menús ajuda al procés de mesura: en primer lloc s'ha d'obrir una imatge (1r menú començant per l'esquerra); després s'ha de calibrar la imatge (2n menú) i després s'ha de mesurar (3r menú).

El menú Mesures s'ha ordenat per ordre alfabètic, atès que no guarden cap tipus de relació entre elles.

La ***Figura 5.2.*** mostra l'aspecte de la barra de menús, amb el primer menú desplegat.

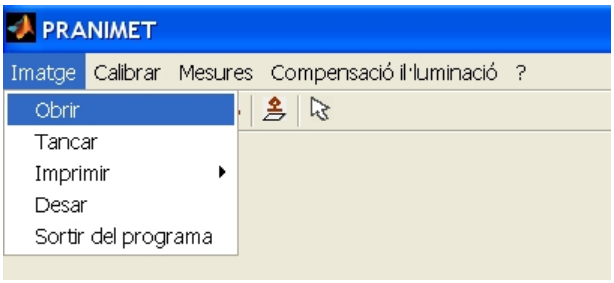


Figura 5.2.. *Aspecte de la barra de menús.*

5.2. Disseny de la barra d'eines

El primer pas consisteix en decidir les eines que es programaran. Tal com s'ha dit anteriorment, les eines hauran de funcionar en totes les pantalles i en totes les mesures. Atenent als requeriments inicials, s'ha decidit incloure les següents:

- Eina de configuració dels colors i dimensions de les línies i del text.
- Eina de nivells automàtics
- Eines de zoom
  - ✓ Zoom in
  - ✓ Zoom out
  - ✓ Pan
- Eines de mesura
  - ✓ Mesura de distàncies o longituds
  - ✓ Mesura de superfícies
- Eina de còpia de la pantalla al portapapers

Aquestes eines s'agrupen en els mateixos 4 grups que estan indicats en el llistat anterior.

El segon pas consisteix en dissenyar les icones i la seva situació. GUIDE no deixa gaire llibertat a l'hora de dissenyar la barra d'eines:

- la col·loca sota la barra de menús
- les eines es situen a partir de l'esquerra
- la icona té una mida estàndard, petita, de 16x16 píxels
- només permet situar un separador a l'esquerra,...

La ***Figura 5.3.*** mostra l'aspecte final de la barra d'eines.

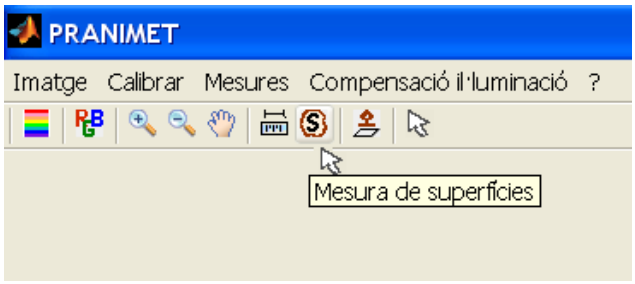


Figura 5.3.. *Aspecte de la barra d'eines.*

5.3. Disseny dels eixos de la imatge

Les imatges que es mesuraran seran imatges en color obtingudes amb una càmera digital. Els formats estàndard en què desen les imatges són:

- relació d'aspecte x/y de 4/3
- relació d'aspecte x/y de 3/2

El primer format és més quadrat que el segon. La relació d'aspecte 4/3 és típica de les càmeres digitals estàndard, mentre que la relació 3/2 és típica de les càmeres analògiques i de les digitals d'alta qualitat.

En el disseny de la interfície s'ha optat per una relació d'aspecte de 4/3, per dues raons:

- Les imatges i la càmera que es disposen en l'actualitat tenen aquesta relació d'aspecte.
- La visualització de la interfície gràfica presenta un aspecte més equilibrat.

De tota manera, no hi ha cap inconvenient en obrir imatges que tinguin una relació d'aspecte 3/2: la imatge no ocuparà tota l'altura de l'*Eix*, i es veurà més allargada.

Atès que la missió del programa és realitzar unes determinades mesures sobre la imatge, aleshores interessa que tingui la màxima resolució. Per tant, es treballarà amb les imatges amb la màxima resolució i no es redimensionaran a la baixa ni es comprimiran.

Per acabar, atès que la imatge s'ha de calibrar i que cal indicar sempre l'escala damunt de totes les imatges, es desactivaran tots els eixos (línies, divisions, escales,...) que dibuixa automàticament MATLAB.

5.4. Disseny dels comandaments

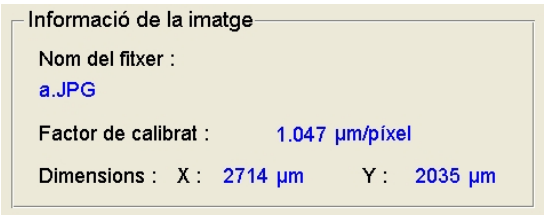
La zona dreta de la pantalla l'ocupen els comandaments.

D'acord amb les normes i restriccions que es donen al començament d'aquest capítol, caldrà fer un conjunt de comandaments per a cada mesura o acció que realitzi el programa. Cada un d'aquest conjunt de comandaments s'englobarà en un panel diferent. D'aquesta manera es podran activar i desactivar a voluntat.

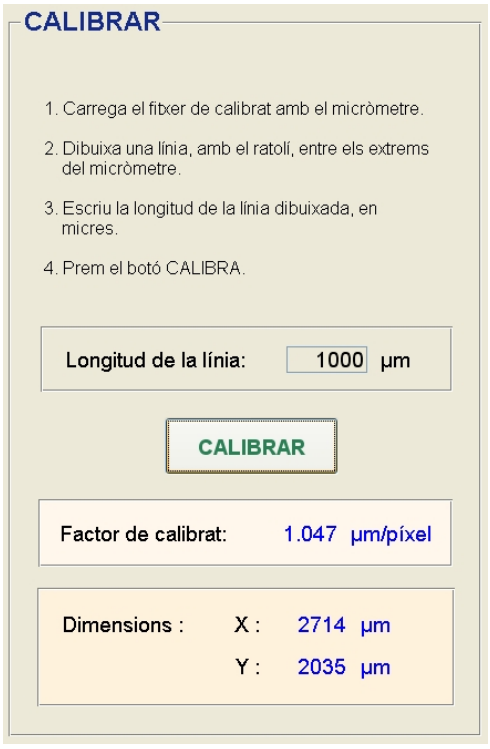
Tenint en compte totes les mesures i altres operacions que ha de realitzar el programa, s'han dissenyat un total de 8 *Panels de comandament*:

1. Panel d'informació de la imatge

Aquest panel és visible sempre, i serveix per donar informació de la imatge (nom de la imatge, factor de calibrat en [µm/píxel], i dimensions de la imatge en [µm]).



2. Panel de calibrat de la imatge



Aquest panel guia el procés del calibrat de les imatges.

Entrades:

- Longitud de la línia

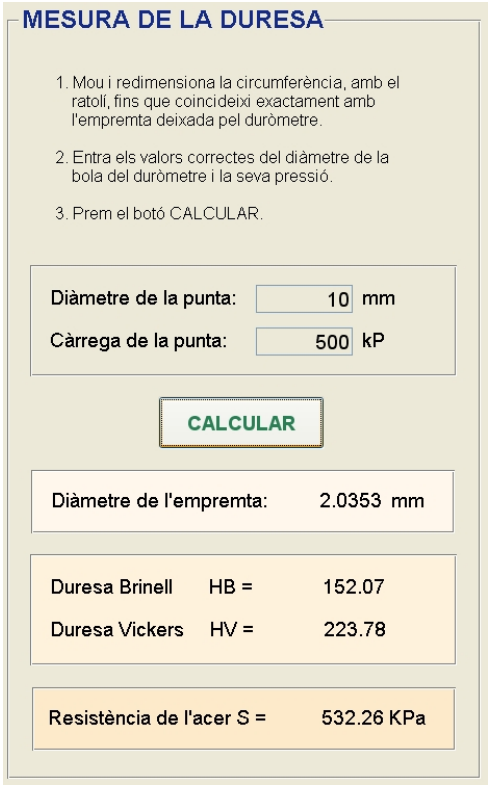
Sortides:

- Factor de calibrat
- Dimensions de la imatge

Comandaments:

- Botó Calibrar

3. Panel de mesura de la duresa



Serveix per guiar el procés de mesura de la duresa Brinell o Vickers.

Entrades:

- Diàmetre de la punta
- Càrrega de la punta

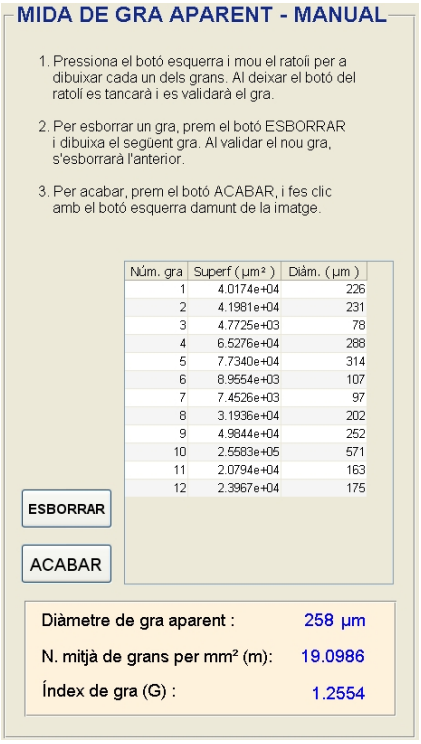
Sortides:

- Diàmetre de l'empremta
- Duresa Brinell
- Duresa Vickers
- Resistència de l'acer

Comandaments:

- Botó Calcular

4. Panel de mida del gra aparent (manual)



Serveix per guiar el procés manual de mesurar la mida aparent del gra.

Entrades:

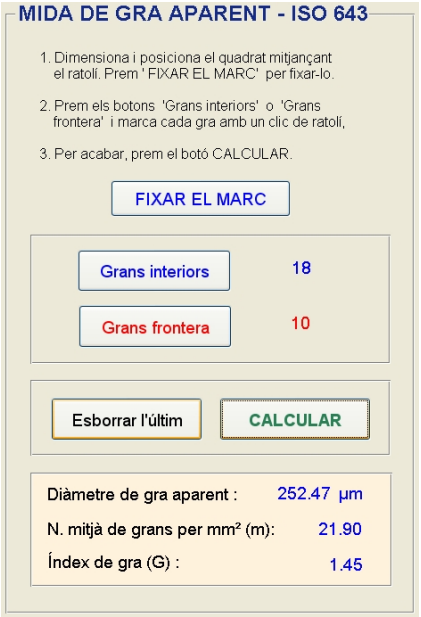
Sortides:

- Taula dimensions dels grans
- Diàmetre aparent del gra
- Nombre de grans per mm²
- Índex de gra

Comandaments:

- Botó Esborrar
- Botó Acabar

5. Panel de mida del gra aparent (ISO 643)



Serveix per guiar el procés de mesurar la mida aparent del gra segons la normativa UNE-EN ISO 643.

Entrades:

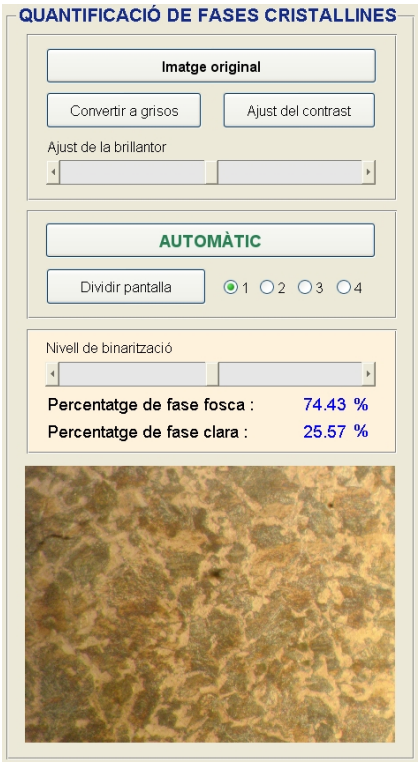
Sortides:

- Diàmetre aparent del gra
- Nombre de grans per mm²
- Índex de gra

Comandaments:

- Botó Fixar el marc
- Botó Grans interiors
- Botó Grans frontera
- Botó Esborrar l'últim
- Botó Calcular

6. Quantificació de fases cristal·lines



Serveix per guiar el procés de quantificar el percentatge de fases cristal·lines en una mostra.

Entrades:

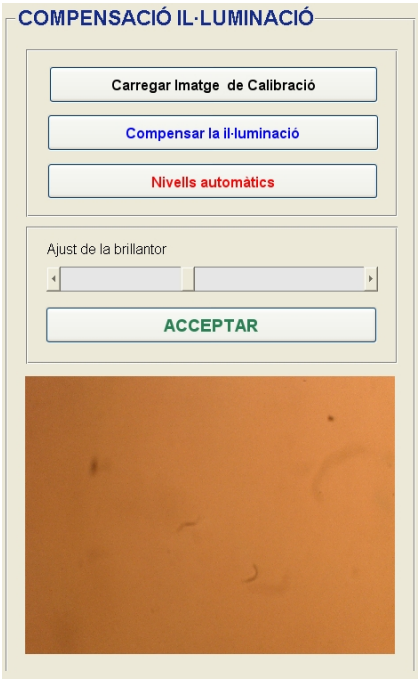
Sortides:

- Percentatge de fase fosca
- Percentatge de fase clara

Comandaments:

- Botó Imatge original
- Botó Convertir a grisos
- Botó Ajust del contrast
- Cursor Ajust de la brillantor
- Botó Automàtic
- Botó Dividir pantalla
- Botons Número de rutina
- Cursor Nivell de Binarització

7. Compensació de la il·luminació



Serveix per guiar el procés de compensar els efectes de la il·luminació sobre les mostres mitjançant una imatge patró.

Entrades:

Sortides:

Comandaments:

- Botó Carregar imatge de calibració
- Botó Compensar la il·luminació
- Botó Nivells automàtics
- Cursor Ajust de la brillantor
- Botó Acceptar

8. Configuració

Serveix per configurar el color, les dimensions i altres paràmetres de totes les línies i els textos.

Entrades:

Sortides:

Comandaments:

- 31 Llistes desplegables
- Botó Tancar panel
- Botó Desar configuració

La idea general per a totes les mesures és crear un panel que tindrà el nom de la mesura que es pretén fer. En aquest panel s'hi situaran tots els comandaments i quadres de text que calgui aplicar. La raó de fer-ho d'aquesta manera és la gran facilitat per activar o desactivar un panel i, per tant, tots els comandaments i quadres de text que contingui. La funció *panels_invisibles* següent posa a *'off'* la propietat '*Visible*' de tots els panels. Això farà que desapareguin tots de la pantalla.

```
function panels_invisibles(handles)
set(handles.info_foto_panel,'Visible','off');
set(handles.calibrar_panel,'Visible','off');
set(handles.mida_gra_panel,'Visible','off');
set(handles.mida_gra_iso_panel,'Visible','off');
set(handles.duresa_panel,'Visible','off');
set(handles.quantificacio_panel,'Visible','off');
set(handles.configuracio_panel,'Visible','off');
set(handles.compensacio_illuminacio_panel,'Visible','off');
```

La següent instrucció 'farà visible' només el panel *info_foto_panel*

```
set(handles.info_foto_panel,'Visible','on');
```

En aquest punt es posa de manifest un dels problemes de GUIDE quan l'aplicació que es dissenya té molts *Panels*: suposant que s'hagi col·locat un *Panel* en una posició determinada de la figura, si es col·loca un altre *Panel* damunt del primer, de manera que estigui totalment inclòs dintre de les dimensions del primer, aleshores el segon *Panel* queda inclòs dintre del primer; això provocarà que actuïn com un sol *Panel*, i no com dos, que és el que es pretenia.

En l'aplicació que s'ha realitzat es situen 8 *Panels* en la mateixa posició de la *Figura*, un sobre l'altre. Això implica que cal ajustar molt bé les dimensions de cada *Panel* (cada *Panel* ha de tenir alguna dimensió més gran que no pas el *Panel* anterior) de manera que no quedin englobats un dintre de l'altre.

Per acabar, cal tenir present que el disseny dels 8 *panels* no és, només, un pur exercici de disseny gràfic sobre el paper (o sobre l'ordinador). De vegades, la necessitat d'incloure un determinat botó de comandament s'ha originat durant la programació d'aquella mesura. Així doncs, el disseny final de cada *Panel* no s'ha fet d'entrada, sinó que a l'inici només s'han marcat les pautes generals del *Panel*, i s'ha anat completant el seu disseny a mesura que s'ha anat avançant en la seva programació.

El disseny de la interfície gràfica s'ha realitzat, exclusivament, amb l'ajuda de l'eina GUIDE de MATLAB.

6. Disseny dels processos de mesura

Ja s'ha comentat en el capítol anterior que la realització d'una aplicació amb l'ajuda de GUIDE és un treball que implica, conjuntament, la utilització de l'eina GUIDE per a crear la interfície gràfica, i la programació clàssica en MATLAB per a generar les instruccions que executaran els comandaments.

El procés és iteratiu:

- En primer lloc es fa un disseny inicial del panel corresponent mitjançant GUIDE.
- A continuació es fa la programació de cada una de les funcions generades en el primer pas. Durant la programació, pot succeir que s'hagi de modificar el disseny original, ja sigui modificant els comandaments, ja sigui afegint-ne de nous.
- Això implicarà utilitzar de nou GUIDE per a modificar la interfície d'usuari.

Aquest procés es repetirà, fins que s'aconsegueixi la funcionalitat desitjada.

En aquest capítol es descriu el disseny dels processos que segueix el programa per a efectuar cada una de les mesures.

Totes les funcions generades es desen en un fitxer clàssic de MATLAB, amb el mateix nom que el fitxer generat per GUIDE, però amb l'extensió *.m*.

Es poden utilitzar totes les instruccions de MATLAB, que són moltes i molt potents.

Disseny dels processos de mesura

Com que la programació en MATLAB no té gaires secrets, l'exposició del disseny dels processos de mesura es fa de la manera següent:

- Es comença per una breu descripció de cada mesura o acció, aprofundint en aquells cassos que es considera que són importants, o en els que la solució aportada ho mereix.
- S'explicita el procés pràctic que s'ha de seguir per efectuar cada mesura o actuació.

- Es llisten les funcions que formen part de cada procediment, amb una breu descripció del seu funcionament.
- Al final de la memòria s'adjunta el llistat del programa.

Les mesures i accions bàsiques que executa el programa són:

a. ***Operacions bàsiques amb fitxers d'imatges.***
b. ***Calibrar una imatge.***
c. ***Mesurar la duresa d'un material.***
d. ***Calcular la mida aparent de gra segons la norma UNE-EN ISO 643.***
e. ***Calcular la mida aparent de gra de manera manual.***
f. ***Quantificar les fases cristal·lines d'una mostra.***
g. ***Compensar la il·luminació del microscopi.***
h. ***Aplicació de les diverses eines.***

6.1. Operacions bàsiques amb fitxers d'imatges

Les operacions bàsiques que es poden fer amb els fitxers d'imatges són les bàsiques de qualsevol aplicació. En aquest cas només s'indiquen les accions que es realitzen:

- Obrir una imatge
- Desar la imatge de treball
- Imprimir la imatge de treball
- Tancar la imatge
- Sortir del programa.

Quan s'engega el programa s'executen les següents accions:

1. Es carreguen els paràmetres per defecte, a partir del fitxer *opcions.mat.*
2. S'amaguen tots els *Panels*
3. Es carrega i visualitza la imatge de portada desada al fitxer portada.png
4. Es crea i s'inicialitza *l'Eix 1*
5. S'inicialitzen les variables generals del programa

Opció de menú Imatge → Obrir

1. Demana el nom de la imatge
2. Amaga la portada i carrega la imatge
3. Si abans s'ha calibrat el sistema, dibuixa l'escala
4. Activa el panel d'informació i actualitza el nom del fitxer carregat.

Opció de menú Imatge → Desar

1. Desa la imatge de la pantalla en un fitxer *.emf*

Opció de menú Imatge → Imprimir → Automàtic

1. Genera una nova figura
2. Hi copia la imatge i tots els altres fitxers gràfics
3. Imprimeix la figura amb els paràmetres per defecte
4. Esborra la figura creada.

Opció de menú Imatge → Imprimir → Manual

1. Engega el Tool d'impressió de MATLAB

Opció de menú Imatge → Tancar

1. Finestra emergent amb petició de confirmació
2. Esborra la imatge
3. Amaga els panels.

Opció de menú Imatge → Sortir del programa

1. Finestra emergent amb petició de confirmació
2. Esborra tots els objectes
3. Tanca el programa.

6.2. Calibrar una imatge

Perquè el programa de mesura pugui mesurar i analitzar correctament les imatges preses amb una càmera fotogràfica cal que es satisfacin 2 requisits:

1. Totes les imatges que s'analitzaran han d'estar preses amb la mateixa càmera, i en les mateixes condicions de resolució i de zoom, tant del microscopi com de la càmera digital. Això significa que totes les imatges tindran la mateixa resolució i les mateixes dimensions.
2. S'ha de fotografiar un micròmetre en les mateixes condicions en què s'han pres les altres fotos. Serà la imatge de calibrat de totes les altres.

Si es satisfan aquestes dues condicions, una vegada s'ha calibrat el programa amb la fotografia del micròmetre, el calibrat és vàlida per a totes les imatges de la sèrie.

El procediment per calibrar consisteix en mesurar quants píxels mesura una longitud coneguda del micròmetre. A partir d'aquest valor es troba el factor de calibrat:

$$Factor\ de\ calibrat = \frac{Longitud\ real\ del\ micròmetre\ (en\ micres)}{Nombre\ de\ píxels\ que\ mesura\ el\ micròmetre} \qquad (6.1)$$

Conegut el factor de calibrat, només caldrà multiplicar la distància mesurada en píxels sobre la imatge per aquest factor i s'obtindrà la mesura de la distància en µm.

Procediment de mesura

El procediment de mesura consisteix en els passos següents:

1. Carregar la imatge de calibrat, amb el micròmetre.
2. Activar el procediment de calibrat a partir de

   Menu → Calibrat → Calibrat

3. Apareix el panel corresponent, i la possibilitat de dibuixar interactivament una línia amb el ratolí.

4. Dibuixar una línia que ocupi una determinada longitud del micròmetre (quan més llarga sigui la línia, més precisió tindrà).

5. Entrar la longitud, en micres, que mesura la línia dibuixada, a partir de la informació que dóna el micròmetre.

6. Prémer el botó ***Calibrar***.

7. El programa calcula el factor de calibrat i les dimensions de la imatge i mostra aquests valors i dibuixa l'escala.

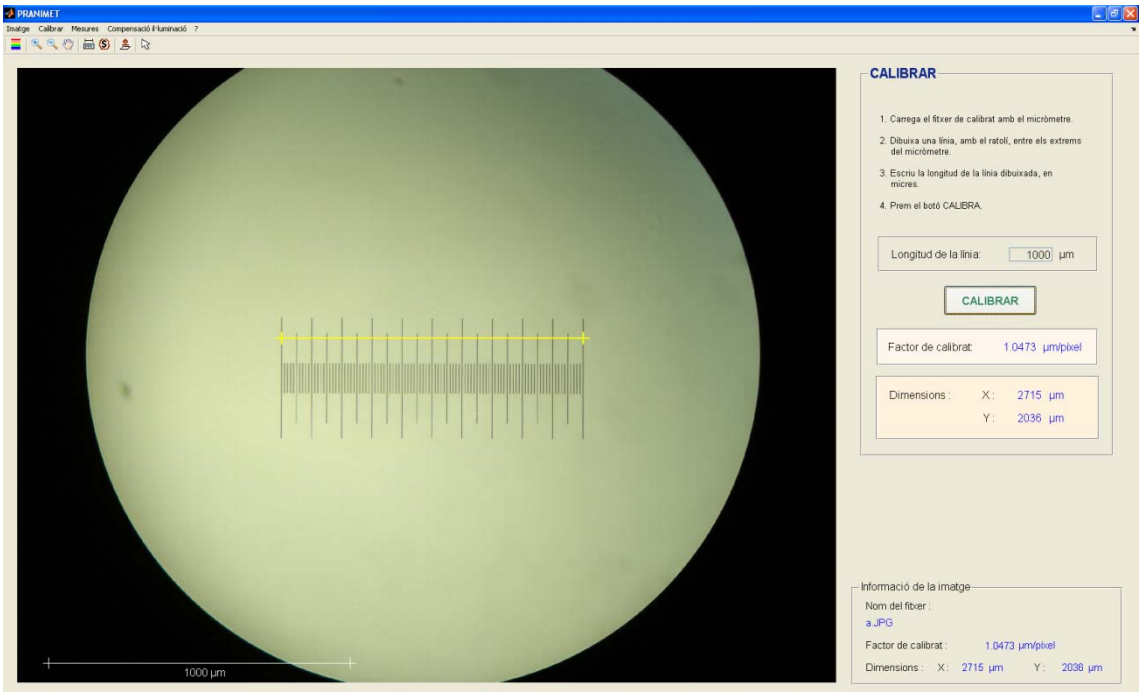


Figura 6.1. *Calibrat de les imatges .*

En el moment de dibuixar la línia, l'usuari pot utilitzar l'eina ***ZOOM*** per ajustar millor la seva posició. Així s'aconsegueix una exactitud millor.

Aquest pas es pot veure en la ***Figura 6.2.***

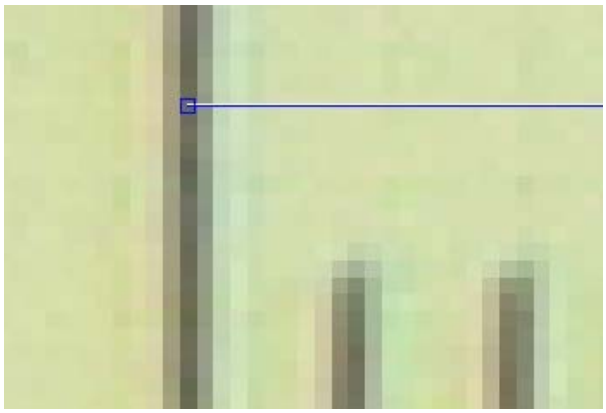


***Figura 6.2**. Utilització del Zoom per millorar el procés de calibrat .*

A continuació es fa una breu descripció de les funcions que composen l'acció de CALIBRAR:

El conveni que s'ha utilitzat per nomenar les funcions és el següent:

1. Si es tracta d'un menú, la funció cridada serà:

   menu_'nomMenu'_'nomSubmenu'_Callback

   Per exemple, si se selecciona el menú *mesures*, dintre d'aquest el submenú *mida de gra aparent*, i dintre d'aquest el submenú *manual*, la funció cridada serà:

   menu_mesures_midagra_manual_Callback

2. Si es tracta d'un comandament (UIcontrol) que pertany a un menú determinat, la funció cridada serà:

   'nomSubmenu'_'UIcontrol'_'nomFunció'_Callback

   Per exemple, la funció *binaritzacio*, que correspon a un cursor (*slider*) i que es troba en el menú *quantificacio*, tindrà el nom següent:

   quantificacio_slider_binaritzacio__Callback

3. Si es tracta d'una funció cridada per una eina de la barra d'eines, o una funció general que pot ser cridada per altres funcions, tindrà el seu nom original. Per exemple, el nom següent correspon a la funció cridada per l'eina *mesura de superfície*:

   mesura_superficie_Clicked_Callback

Descripció de les funcions

menu_calibrar_calibrar_Callback

- Mostra el panel calibrar
- Espera que l'usuari dibuixi la línia sobre el micròmetre

calibrar_edit_micres_CreateFcn

- Neteja el camp d'entrada de dades de longitud de línia

calibrar_edit_micres_Callback

- Gestiona l'entrada de dades de la longitud de línia (missatge d'error en cas que no passi el filtre)

calibrar_pushbutton_calibra_Callback

- Mesura la línia dibuixada per l'usuari
- Calcula l'escala
- Actualitza els paràmetres referits a escala i dimensions
- Crida la rutina que dibuixa l'escala
- Actualitza els valors del panel d'informació de la imatge

dibuixar_escala

- Dibuixa l'escala sobre la imatge

6.3. Mesurar la duresa d'un material

A partir de les empremtes Brinell i Vickers deixades pel duròmetre Hoytom sobre la mostra, s'ha de mesurar el diàmetre de les primeres i les diagonals de les segones per calcular la duresa dels materials.

Les empremtes Brinell són circulars i cal passar obligatòriament pel centre per tenir una mesura fiable del seu diàmetre. Una vegada conegut el diàmetre de l'empremta i els paràmetres del duròmetre, es pot calcular la duresa del material:

$$Duresa\ [HB] = \frac{2P}{\pi D\left[D - \sqrt{D^2 - d^2}\right]} \tag{6.2}$$

on *D és el diàmetre de la punta del duròmetre*
d és el diàmetre de l'empremta
P és la pressió o la càrrega del duròmetre

Les empremtes Vickers són romboides i cal mesurar les diagonals.

$$Duresa\ [HV] = \frac{1.854P}{d_1^2} \tag{6.3}$$

on d_1 *és la diagonal de l'empremta*
P és la pressió o la càrrega del duròmetre

Pel cas dels acers també caldria calcular la seva resistència (σ_U)

$$\sigma_u\ (MPa) = 3.5\ HB \tag{6.4}$$

Una manera fiable de fer el càlcul consisteix en ajustar un cercle a les empremtes, i calcular el seu diàmetre.

Procediment de mesura

El procediment de mesura consisteix en els passos següents:

1. Carregar la imatge amb l'empremta deixada pel duròmetre.
2. Activar el procediment de mesura a partir de
   Menu → Mesures → Duresa
3. Apareix el panel corresponent i un cercle al mig de la imatge
4. Arrossegar i redimensionar el cercle, amb el ratolí, de manera que ocupi exactament l'empremta deixada pel duròmetre.
5. Entrar el valor del diàmetre i de la càrrega de la punta sobre la proveta.
6. Prémer el botó ***Calcular.*** El programa calcula la duresa i la resistència de l'acer i mostra aquests valors en el panel.

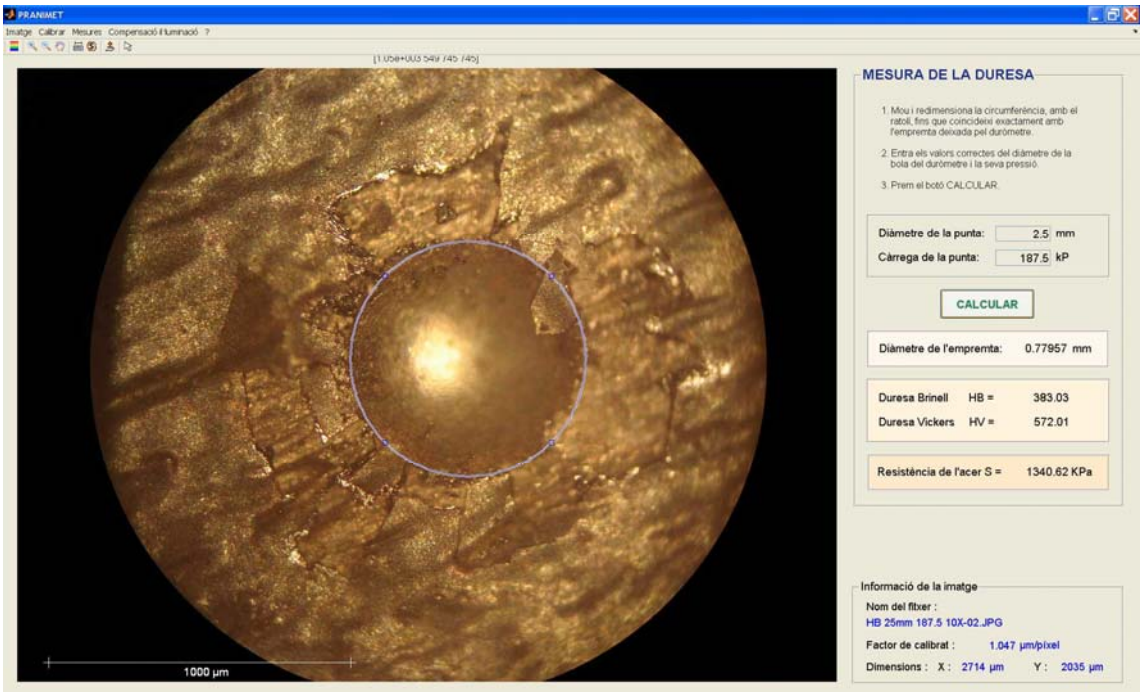


Figura 6.3. *Mesura de la duresa Brinell.*

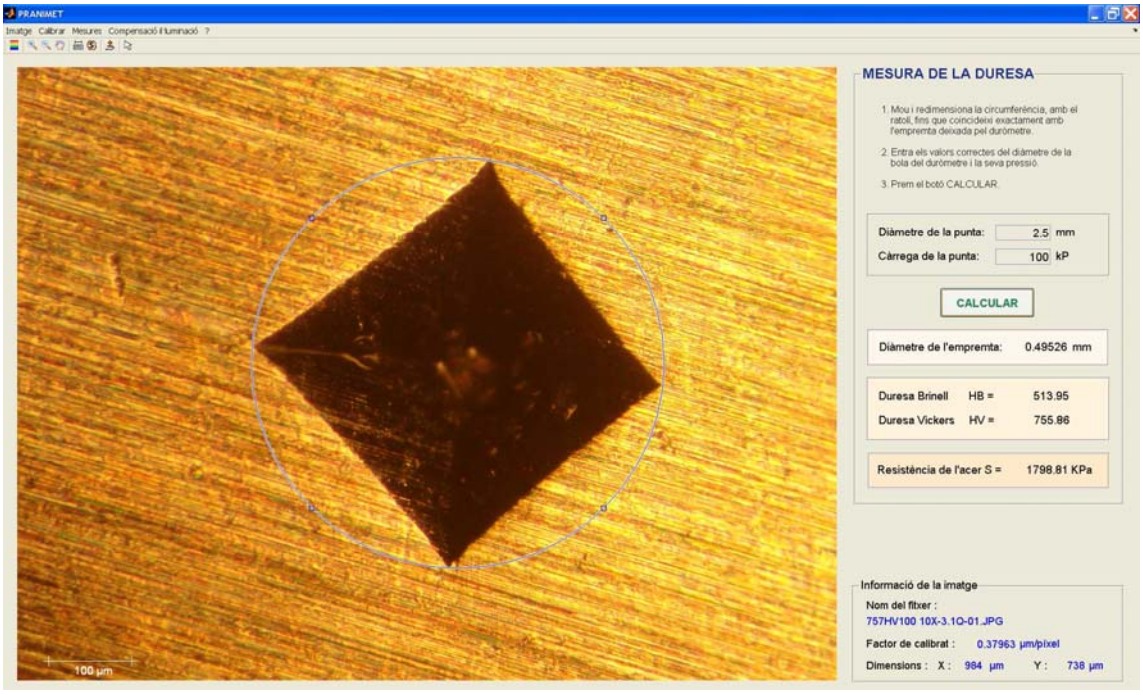


Figura 6.4. *Mesura de la duresa Vickers.*

Descripció de les funcions

menu_mesures_duresa_Callback

- Mostra el panel duresa.
- Dibuixa l'escala.
- Dibuixa un cercle. L'usuari mou i redimensiona el cercle fins a fer-lo coincidir amb els marges de l'empremta del duròmetre.

duresa_edit_diametre_punta_CreateFcn

- Fixa condicions inicials (diàmetre punta = 10mm).

duresa_edit_pressio_punta_CreateFcn

- Fixa condicions inicials (pressió punta = 500KP).

duresa_pushbutton_calcular_Callback

- Gestiona l'entrada de dades del diàmetre de la punta (missatge d'error en cas que no passi el filtre).
- Gestiona l'entrada de dades de la pressió de la punta (missatge d'error en cas que no passi el filtre).
- Mesura el diàmetre del cercle.
- Calcula els valors de la duresa Brinell, Vickers i de la resistència de l'acer.
- Actualitza els valors del panel.

6.4. Calcular la mida aparent de gra segons la norma UNE-EN ISO 643

La norma UNE-EN ISO 643 indica uns quants procediments per calcular la mida aparent de gra d'una mostra, i el seu índex de gra, o índex G.

El procediment que té més exactitud consisteix en dibuixar un quadrat de dimensions conegudes sobre la mosta, i comptar els grans que són dintre del quadrat. Com que hi ha grans que queden dividits per la vora del quadrat, es pren la convenció que els grans totalment interiors puntuen 1, els grans que estan part a dins i part a fora puntuen 0.5, i els 4 grans que estan als quatre angles puntuen, entre tots quatre, 1.

Així doncs, el número total de grans serà:

$$Núm\ grans = N.grans\ interiors + \frac{N.grans\ frontera}{2} + 1 \qquad (6.5)$$

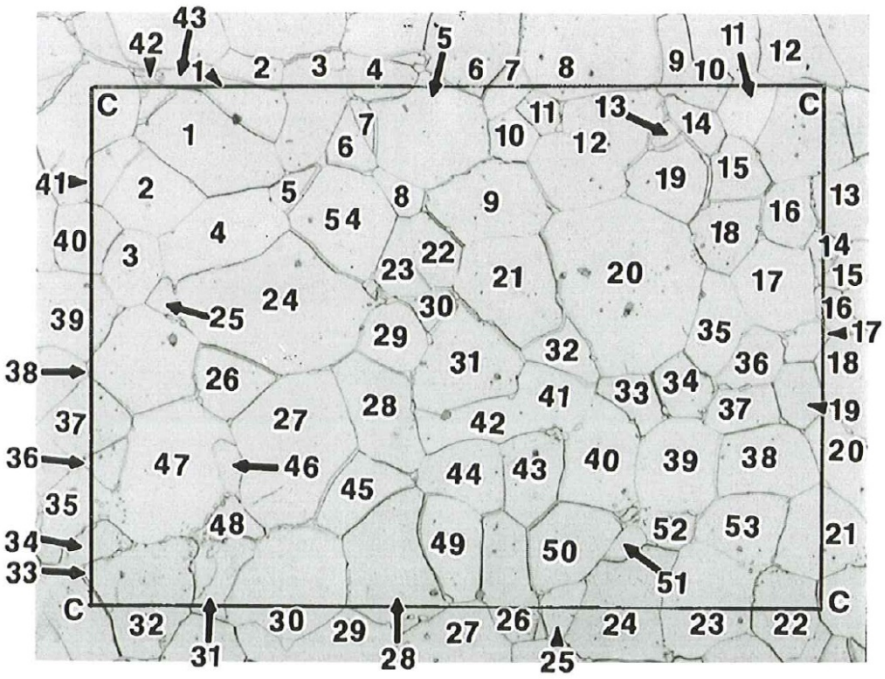


***Figura 6.5**. Comptabilització dels grans segons la norma ISO 643.*

A partir d'aquí es pot trobar el paràmetre m (quantitat de grans per mm²) dividint el Núm. de grans per la superfície del quadrat

$$m = \frac{Núm.\ grans}{Superfície\ quadrat\ [mm^2]} \qquad (6.6)$$

Finalment, l'índex de gra (G) es troba a partir de la següent equació:

$$G = \frac{\log m}{\log 2} - 3 \qquad (6.7)$$

Procediment de mesura

El procediment de mesura consisteix en els passos següents:

1. Carregar la imatge que s'ha de mesurar.
2. Activar el procediment de mesura a partir de

   Menu → Mesures → Mida de gra aparent → UNE-EN ISO 643

3. Apareix el panel corresponent i un quadrat al mig de la imatge
4. Arrossegar i redimensionar el quadrat, amb el ratolí, de manera que englobi uns quants grans al seu interior. La norma ISO parla d'uns 40 o 50.
5. Prémer el botó ***Fixar el Marc***.
6. Prémer el botó ***Grans interiors***, i anar marcant tots els grans que estiguin totalment a l'interior del quadrat.
7. Prémer el botó ***Grans frontera***, i anar marcant tots els grans que estiguin parcialment a l'interior. No marcar els 4 grans que estan en les 4 cantonades.
8. Es pot anar alternant entre ***Grans interiors*** i ***Grans frontera***, i també es poden esborrar els últims grans marcats prement el botó ***Esborrar***.
9. Quan s'ha acabat de marcar els grans, prémer el botó ***Calcular***. El programa calcula el nombre de grans per mm^2, l'índex G i el diàmetre aparent de gra, i els mostra al panel.

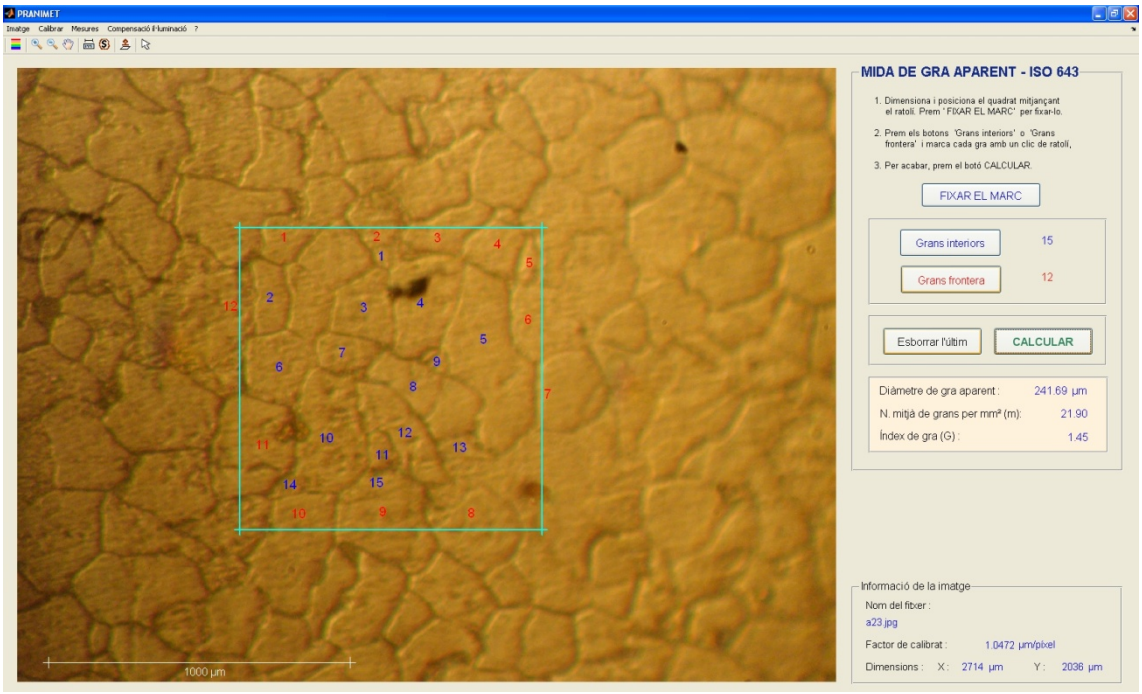


***Figura 6.6**. Mesura de la mida aparent de gra segons la ISO 643.*

Descripció de les funcions

menu_mesures_mida_gra_iso_Callback

- Mostra el panel mida gra iso.
- Posa tots els resultats a zero, i inicialitza tots els comptadors.
- Dibuixa un quadrat interactiu

iso_mesura_grans_pushbutton_fixer_marc_Callback

- L'usuari mou i redimensiona el quadrat.
- Quan l'usuari prem el botó Fixar el Marc, la funció llegeix la posició del quadrat i en dibuixa un de nou, fix.

iso_mesura_grans_pushbutton_interior_Callback

- Es crida aquesta rutina quan l'usuari prem el botó Grans Interiors. A continuació s'inclou el codi d'aquesta rutina, com a exemple de programació en MATLAB:

```
% --- Executes on button press in iso_mesura_grans_pushbutton_interior.
function iso_mesura_grans_pushbutton_interior_Callback(hObject, eventdata,
handles)
global bucle_i;    % Passa variables globals a la rutina
global bucle_f;
global num_gra_interior;
global surt;
global hti;
global f;
global c;
global op;
bucle_i=1;
bucle_f=0;
while bucle_i==1    % Fes, mentre no premis el botó Grans Frontera
   bucle_f=0;      % Inactiva la funció Grans Frontera
   try
      k = waitforbuttonpress;    % Espera que l'usuari premi el ratolí
   catch me
   end
   if surt==1   % Acaba, si s'ha premut el botó Calcular
      break
   end
   point1 = get(gca,'CurrentPoint');    % Detecta la pulsació del botó del ratolí
   point1 = point1(1,1:2);              % Extreu les coordenades del punt
   if point1(1)>=0 & point1(1)<=c & point1(2)>=0 & point1(2)<=f
      if num_gra_interior>9        % Comprova que el ratolí no estigui fora de
         point1(1)=point1(1)-20;   % la imatge.
      else
         point1(1)=point1(1)-10;
      end
      % Crea un nou objecte gràfic, el text corresponent al número del
      % nou gra, i el mostra per la pantalla en la posició del ratolí.
      % Al mateix temps, desa el handle d'aquest objecte en una matriu,
      % per si cal esborrar-lo

hti(num_gra_interior)=text(point1(1),point1(2),sprintf('%0.f',num_gra_interior
),...
          'Color',op(10).c,'FontSize',op(10).w,'Tag','text_mesura_interior');

set(handles.iso_mesura_grans_text_interiors,'String',sprintf('%d',num_gra_int
erior));
      num_gra_interior=num_gra_interior+1;   % Incrementa el número del gra
   end
end
```

iso_mesura_grans_pushbutton_frontera_Callback

- Es crida aquesta rutina quan l'usuari prem el botó Grans Frontera. És simètrica a l'anterior, i són excloents. Quan una s'activa, l'altra s'ha de desactivar.

iso_mesura_grans_pushbutton_esborrar_Callback

- Detecta si s'està treballant amb Grans Interiors o Frontera.
- Esborra l'últim objecte dibuixat (el número de l'últim gra) i decrementa el comptador de grans.

iso_mesura_grans_pushbutton_calcular_Callback

- Calcula la superfície del quadrat.
- Calcula tots els altres paràmetres, i els mostra pel panel.

6.5. Calcular la mida aparent de gra de manera manual

La pròpia norma UNE-EN ISO 643 reconeix que amb els nous computadors es poden utilitzar altres mètodes més exactes per a calcular la mida aparent de gra d'una mostra, i el seu índex de gra, o índex G.

En el programa s'ha inclòs una rutina més exacta, que consisteix en mesurar manualment la superfície de cada gra. D'aquesta manera el resultat és més exacte.

S'ha intentat automatitzar el procés de reconèixer la forma dels grans, però la presència de macles i la poca qualitat de les imatges ho ha fet impossible. Així doncs, el procés es fa de manera manual, i es deixa oberta la porta a aconseguir una rutina que ho aconsegueixi automàticament.

En la ***Figura 6.7*** es pot veure el procés:

1. Mitjançant el ratolí es dibuixa el contorn d'un gra.
2. En el moment que s'allibera el botó del ratolí, la figura es tanca, el programa calcula la seva superfície, marca el gra amb un número correlatiu i calcula automàticament el seu diàmetre aparent i el del conjunt de tots els grans analitzats fins aquell moment.

***Figura 6.7**. Procés de mesura de la mida aparent de gra amb el mode normal.*

Per calcular la superfície de cada gra s'utilitzen funcions específiques de MATLAB:

Quan es tanca una figura, es crea una màscara amb la mateixa forma de la figura. Aquesta màscara és una matriu binària, de manera que tota la matriu està a zeros, excepte la màscara (amb la forma de la figura) que està a uns. Comptant tots els uns, i multiplicant el resultat per la superfície d'un píxel s'obté la superfície de la figura de manera molt exacta.

```
try
   hFH = imfreehand;  % Permet resseguir un gra a mà alçada
catch me              % amb el ratolí
   break
end
if bucle~=1
   break
end
setColor(hFH,[0 1 0]);
binaryImage = hFH.createMask();  % Crea la màscara binària
% Suma tots els '1' i calcula la superfície total del gra
sup_gra=sum(sum(binaryImage==1))*sup_unitaria_pixel;
taula_grans(num_gra,1)=num_gra;  % Crea una nova fila a la taula
taula_grans(num_gra,2)=sup_gra;
% Calcula el diàmetre aparent del gra
taula_grans(num_gra,3)=round(2*(sup_gra/pi)^0.5);
```

Procediment de mesura

El procediment de mesura consisteix en els passos següents:

1. Carregar la imatge que s'ha de mesurar.
2. Activar el procediment de mesura a partir de

   Menu → Mesures → Mida de gra aparent → Mètode manual

3. Apareix el panel corresponent i el programa es posa a disposició de que l'usuari dibuixi els grans.
4. Amb el botó del ratolí premut, l'usuari dibuixa el gra.
5. En el moment que s'allibera el botó del ratolí, el gra es tanca.
6. A continuació, l'ordinador calcula la superfície i el diàmetre aparent del gra, afegeix els paràmetres del gra a la taula del panel, afegeix un número d'índex al gra, calcula els valors d'índex de gra i diàmetre mig per al conjunt de grans, i es posa a disposició de dibuixar un nou gra.
7. Si es prem el botó ***Esborrar***, el programa recupera l'última imatge abans de dibuixar el nou gra, i elimina el gra de la taula, recalculant de nou tots els valors.
8. En el moment que es prem el botó ***Acabar***, el programa acaba definitivament la rutina.

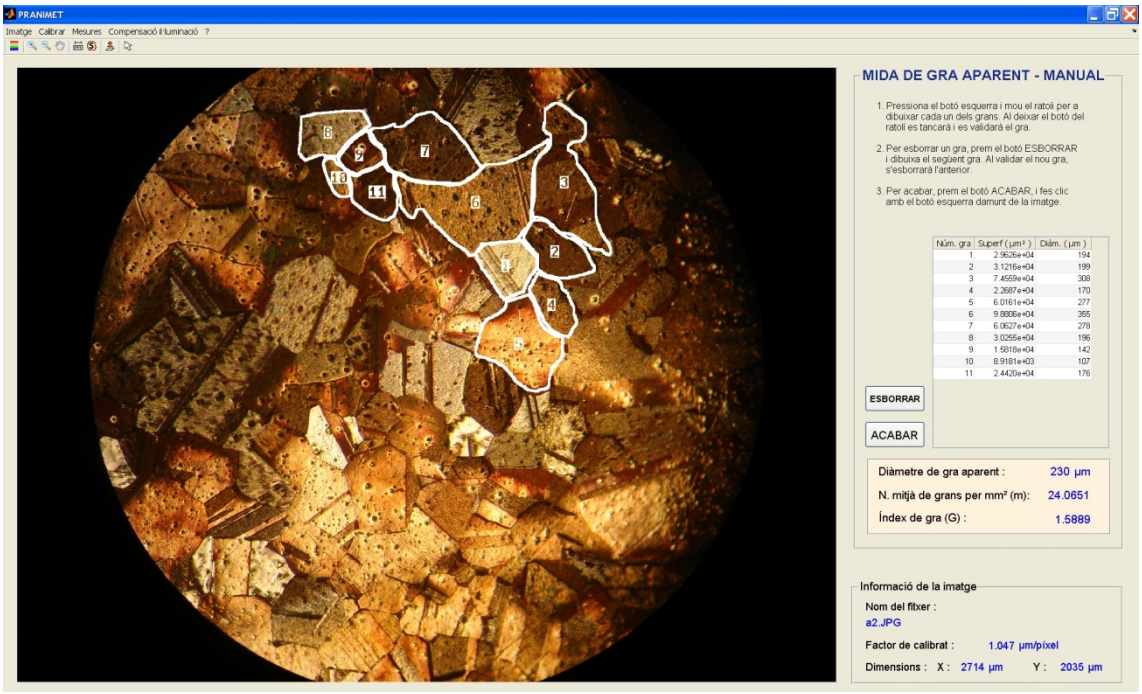


***Figura 6.8**. Mesura de la mida aparent de gra amb el mètode manual.*

Descripció de les funcions

menu_mesures_mida_gra_manual_Callback

- Mostra el panel mida gra manual.
- Posa tots els resultats a zero, i inicialitza tots els comptadors.
- Permet dibuixar un gra a mà alçada, amb el ratolí.
- Genera la màscara.
- Calcula la superfície del gra.
- Dibuixa la màscara i el número de gra sobre la imatge, i la desa la anterior en una matriu per si cal recuperar-la més endavant.
- Actualitza la taula.
- Calcula els resultats
- Actualitza els resultats en el panel.

mida_gra_pushbutton_esborrar_Callback

- Recupera la imatge anterior (sense l'últim gra dibuixat)
- Decrementa en 1 el número de grans.
- Recalcula tots els resultats
- Actualitza la taula i el panel.

mida_gra_pushbutton_acabar_Callback

- Tanca imatges i handles i surt de la rutina.

mida_gra_taula_1_CreateFcn

- Inicialitza la taula de característiques dels grans.

text2im

- Funció que converteix un text d'entrada en una imatge, de manera que es pugui gravar sobre una matriu.

6.6. Quantificar les fases cristal·lines d'una mostra

De vegades és important poder quantificar el percentatge de cada un dels dos (o més) components de què està format un material. Per exemple, podem necessitar saber el percentatge de grafit que hi ha en una fosa, de perlita en un acer, de cuprita en el coure, etc.

Per poder mesurar aquest percentatge cal que el material estigui format bàsicament per dos components, i que aquests components tinguin propietats òptiques el més diferenciades possible. Això ens permetrà fixar un llindar de manera relativament fàcil, i binaritzar la imatge.

Una vegada que la imatge estigui binaritzada correctament, el procés de quantificar les fases es limita a comptar la quantitat d'uns i de zeros que té en total.

En tot aquest procés que s'ha explicat hi ha una operació clau: la binarització de la imatge. Binaritzar una imatge consisteix en transformar una imatge que està composta por molts tons de gris en una imatge en blanc i negre, és a dir, una imatge formada només per zeros (color negre) i per uns (color blanc).

El procés matemàtic que ho fa és molt senzill. Només cal establir un valor llindar (recordem que en una imatge de tons de gris, el valor de cada punt pot anar, per exemple, entre 0 (punt negre) i 255 (punt blanc)). Una vegada fixat el valor llindar (suposem 100), tots els punts que tinguin un valor superior a 100 passaran a valer 1 (color blanc), mentre que tots els punts que tinguin un valor igual o menor que 100 prendran el valor 0 (color negre).

A partir d'aquí és fàcil comptabilitzar quants zeros i quants uns hi ha en una imatge, i quantificar les seves proporcions respectives.

Problemes a l'hora de binaritzar una imatge

L'ajust més important que s'ha de fer al binaritzar una imatge és la tria del valor llindar. En la imatge següent es poden veure tres fotografies: la primera correspon a la imatge original, mentre que les altres dues són binaritzacions de l'original, amb dos valors de llindar diferents. Es pot veure que el resultat canvia molt.

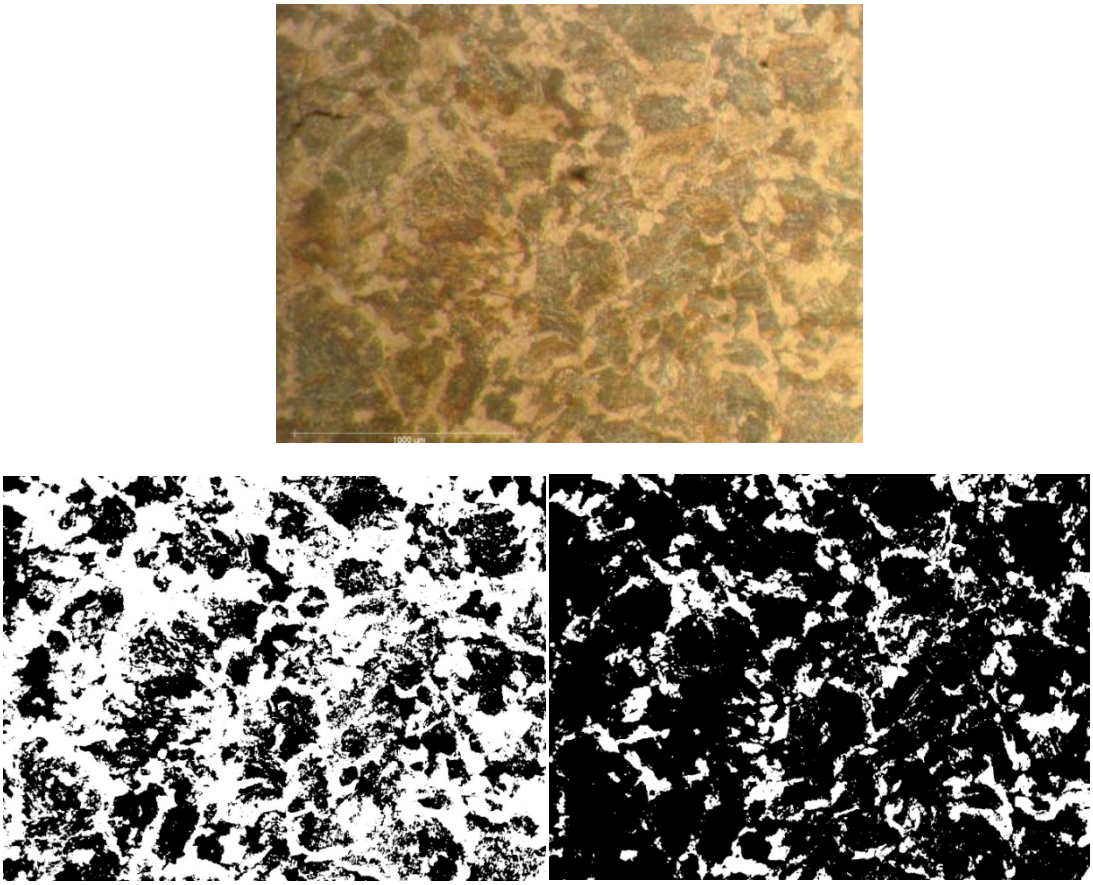


***Figura 6.9**. Efecte de variar el llindar en un procés de binarització.*

En la imatge de l'esquerra, el nivell de binarització és baix, i això fa que tots els grisos es converteixin en blancs. En aquesta fotografia el percentatge de fase fosca respecte del total no passa del 42%.

En la imatge de la dreta, en canvi, el nivell de binarització és alt, i això fa que tots els grisos es converteixin en negres. En aquesta fotografia el percentatge de fase fosca respecte del total supera el 79%.

És molt difícil programar una rutina que encerti automàticament el valor del llindar. Dependrà molt de com siguin les dues fases que estan presents en el material, i de com estigui presa la fotografia. Habitualment es fixa un valor llindar situat en la zona mitja, i es deixa que sigui el propi observador que l'acabi d'ajustar per comparació amb la imatge original.

Això és el que fa la rutina que s'ha implementat: converteix la imatge a tons de gris, i després la binaritza a partir d'un llindar obtingut de la mitja de tots els valors de la imatge. Una vegada binaritzada la imatge, l'usuari pot acabar d'ajustar el llindar comparant la imatge binaritzada amb l'original. Per aquest motiu s'ha dissenyat la interfície de manera que pugui mostrar les dues imatges, una al costat de l'altra, i poder-les comparar més fàcilment.

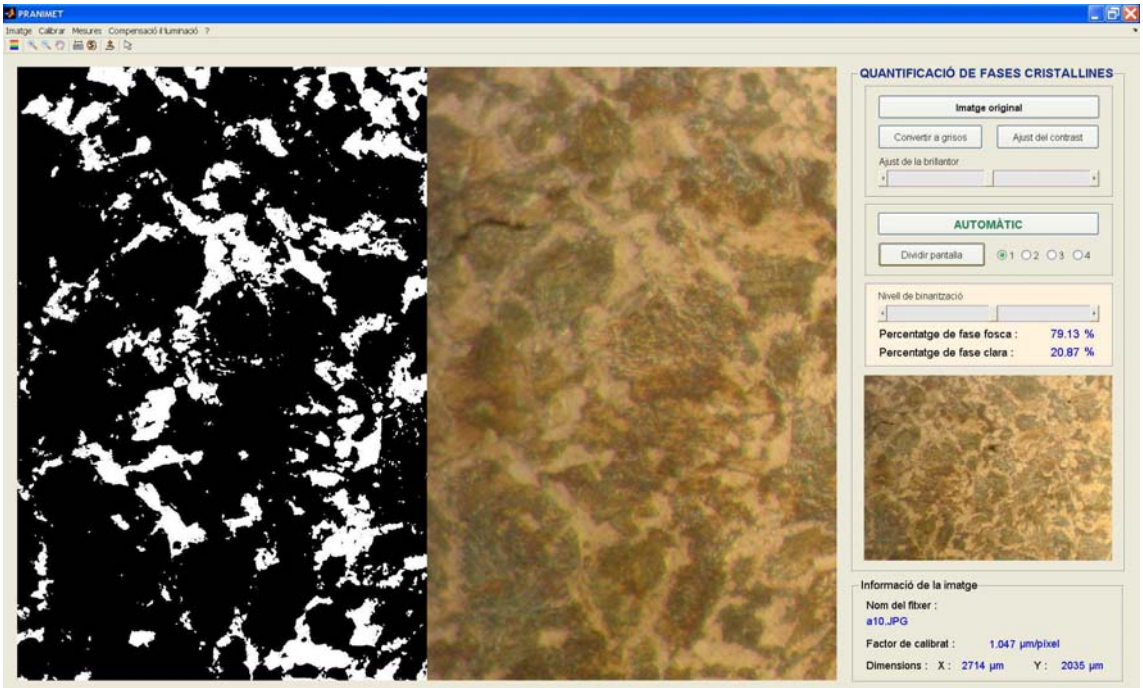


***Figura 6.10**. Binarització de la imatge: comparació per ajustar el llindar.*

En la ***Figura 6.10*** es mostra una imatge del procés de binarització, i es veu com al posar les imatges una al costat de l'altra es poden comparar millor. En aquesta imatge es veu que el valor del 79% de fase fosca s'assembla molt més a la realitat que no pas el 42%.

Un altre problema important que apareix es deu als efectes d'una mala il·luminació del microscopi que no presenti una intensitat homogènia en tot el camp de visualització. Això provoca zones més clares i zones més fosques en la imatge que, a l'hora de binaritzar-les provocarà que sigui impossible fer-ho bé. En la figura següent es mostra una imatge real obtinguda amb un microscopi amb la il·luminació mal ajustada, i la seva binarització:

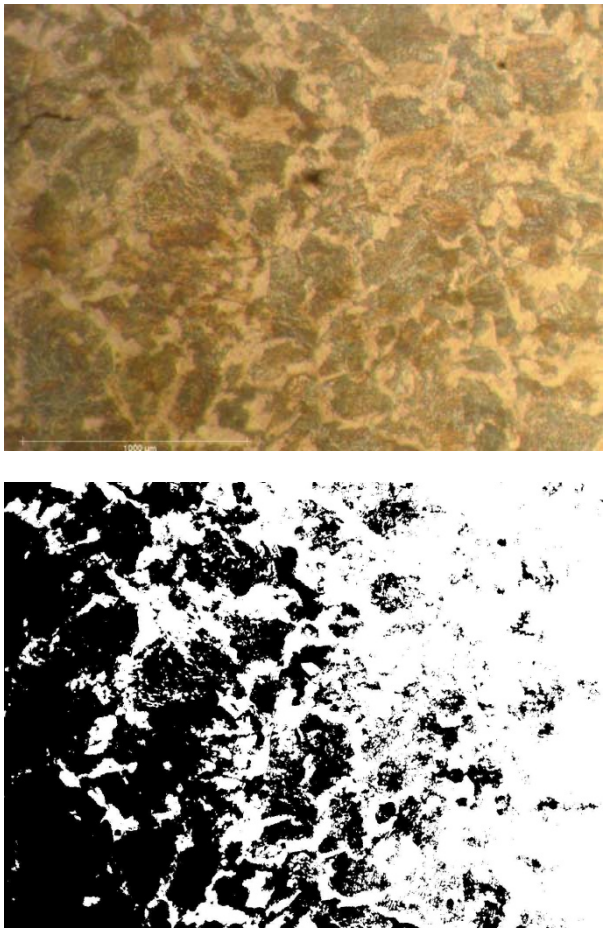


***Figura 6.11**. Problema en la binarització degut a la mala il·luminació.*

Es veu clarament que la zona superior dreta de la imatge està molt més il·luminada que la resta. Això provoca que si s'ajusta el llindar d'acord amb la part superior dreta de la imatge, la part inferior esquerra es veurà completament negra, i a l'inrevés.

Una solució a aquest problema és la compensació mitjançant una foto patró que es fa del microscopi sense cap mostra, només de la il·luminació. La teoria diu que dividint la imatge original per la imatge patró, el resultat queda independent de la

il·luminació. Aquest procediment s'ha implementat en el programa i s'explica en l'apartat següent.

De tota manera, i per les raons que sigui (segurament les fotografies es varen prendre en moments distants en el temps; poden haver variat les condicions ambientals d'il·luminació, etc), el resultat no és tan bo com diu la teoria. En la ***Figura 6.12*** es veu la imatge original compensada mitjançant aquest mètode, i el resultat de binaritzar-la.

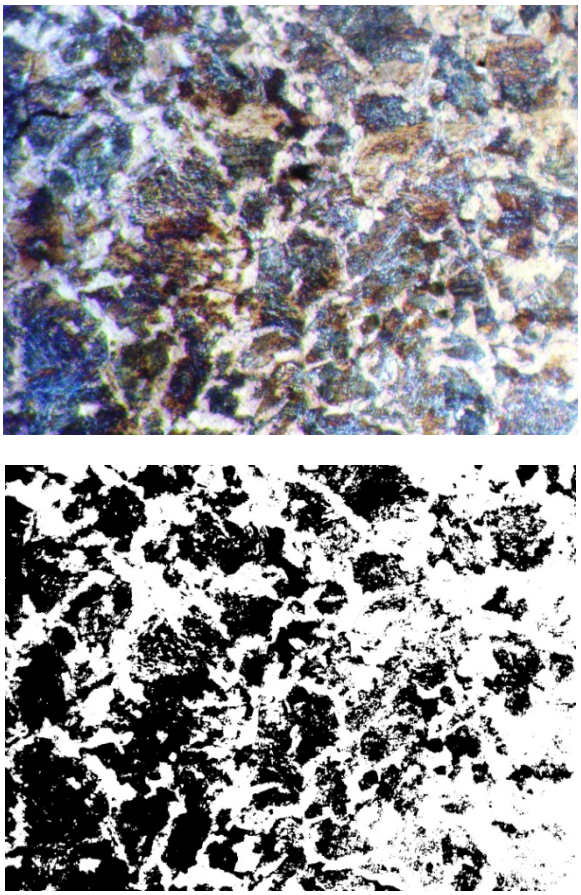


Figura 6.12*. Binarització amb compensació de la il·luminació.*

Es pot veure que s'ha guanyat qualitat respecte la de la ***Figura 6.11***, però el resultat encara no és prou bo.

Compensació automàtica de la il·luminació

Per solucionar aquest problema s'ha realitzat una rutina que efectua una compensació automàtica de la il·luminació.

1. El procediment comença suposant que les fases estan repartides de manera més o menys homogènia pel material, i es converteix la imatge en tons de gris.

2. A continuació es divideix la imatge en 12 franges verticals completament equidistants (la quantitat de franges és un paràmetre que es pot canviar), i es comptabilitza la suma de tots els punts de cada franja.

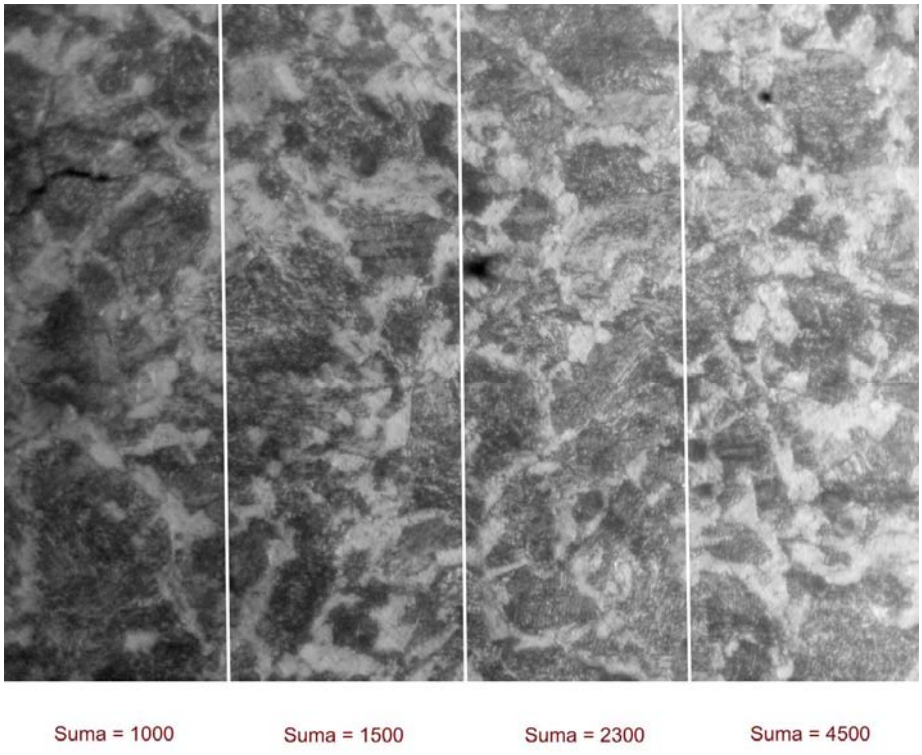


***Figura 6.13**. Procediment per a compensar la il·luminació.*

   En aquest exemple la imatge s'ha dividit, només, en 4 franges, i la suma dels valors de cada franja és inventat.

3. A partir d'aquí es calculen les diferències entre cada franja i la primera de l'esquerra.

En l'exemple anterior:

F1	F2	F3	F4
0	+500	+1300	+3500

4. A continuació es calcula el valor que s'ha de sumar o restar a cada punt de cada franja (excepte la primera), perquè la seva suma total sigui igual a la de la franja de l'esquerra.

   En l'exemple anterior, suposem que el càlcul hagi donat

F1	F2	F3	F4
0	-15	-35	-50

   Això vol dir que si a cada punt de la franja 2 se li resta 15, quan es faci la suma de tots els punts de la franja 2 donarà 1000, igual que la franja 1. I això passarà igual en totes les altres franges.

5. A continuació es construeix un vector de compensació que té la mateixa amplada que la imatge, en el que els valors de les posicions equivalents als punts centrals de cada franja valen, precisament, 0, -15, -35 i -50. Entremig d'aquestes posicions centrals, els valors s'interpolen linealment.

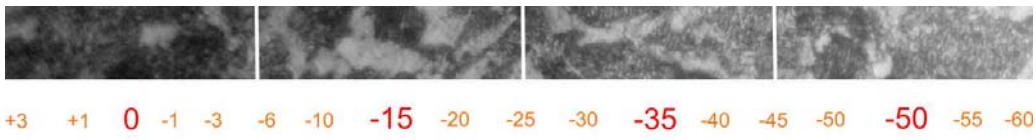


***Figura 6.14**. Construcció del vector de compensació.*

6. A continuació es genera una matriu de compensació replicant aquest vector tantes vegades com files tingui la imatge original.

7. El següent pas és sumar la matriu de compensació a la imatge original. Amb aquesta operació ens assegurem que totes les franges tindran aproximadament el mateix valor promig, i que la transició serà lineal i contínua al llarg de cada franja i de tota la imatge.

8. A continuació es divideix la imatge en 8 franges horitzontals i es repeteix tot el procés anterior, prenent columnes per files i a l'inrevés. El resultat final és una imatge en la que la *densitat* de valors és constant, globalment, tant en l'eix d'ordenades com en el d'abscisses.

El resultat d'aplicar aquesta rutina es mostra en la ***Figura 6.15***. Es veu que el resultat de la compensació és molt bo, millor que la compensació per imatge patró que es veurà en el proper apartat.

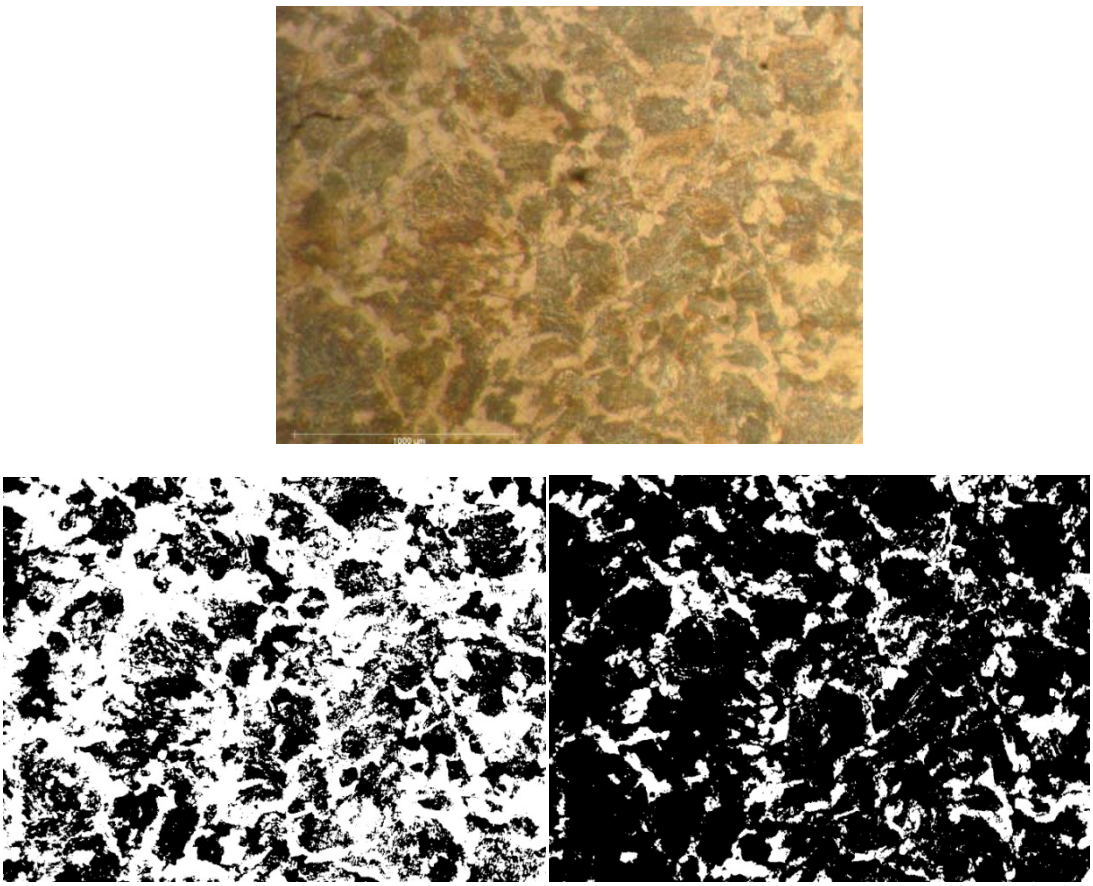


Figura 6.15*. Resultat de la compensació automàtica de la il·luminació.*

Per acabar amb la fase de disseny d'aquesta mesura cal comentar que, a més a més de la compensació automàtica, també s'ha implementat el procediment per a efectuar la binarització de manera totalment manual.

Procediment de mesura

En el paràgraf anterior s'ha comentat que s'han implementat dos procediments diferents per a realitzar la mesura: el manual i l'automàtic.

El procediment manual de mesura consisteix en els passos següents:

1. Carregar la imatge que s'ha de mesurar.

2. Activar el procediment de mesura a partir de

Menu → Mesures → Quantificació de fases cristal·lines

3. Apareix el panel corresponent i en el mateix panel apareix una còpia de mida petita de la imatge original, amb l'objectiu de poder comparar els passos que es van fent amb la imatge original.

4. Al prémer el botó ***Convertir a grisos***, la imatge es converteix a escala de grisos.

5. Al prémer el botó ajust del contrast, apareix una finestra volada amb la que es poden ajustar manualment els nivells de la imatge. En general s'arrosseguen els dos cercles vermells fins que la ratlla divisòria coincideixi amb l'inici i el final de l'histograma. Ajustat el contrast, es tanca la finestra volada.

6. En cas de necessitat es pot utilitzar el cursor ***Ajust de la brillantor*** per ajustar la brillantor general.

7. El pas final és ajustar el llindar de la binarització mitjançant el cursor ***Nivell de binarització***. Prement el botó ***Dividir pantalla*** s'activa/desactiva l'opció de dividir la imatge per la meitat i mostrar la imatge original, amb la finalitat d'ajustar millor el nivell de binarització. El botó ***Imatge original*** permet descartar els passos fets i recuperar la imatge original.

8. En tot moment es poden llegir els percentatges de fase clara i fosca en el panel.

El procediment automàtic de mesura consisteix en els passos següents:

1. Carregar la imatge que s'ha de mesurar.

2. Activar el procediment de mesura a partir de

Menu → Mesures → Quantificació de fases cristal·lines

3. Apareix el panel corresponent i en el mateix panel apareix una còpia de mida petita de la imatge original, amb l'objectiu de poder comparar els passos que es van fent amb la imatge original.

4. Al prémer el botó ***AUTOMÀTIC*** s'aplica la rutina que compensa automàticament la il·luminació de la imatge. A partir d'aquí el procés continua en el punt 7) del procediment manual que s'ha explicat anteriorment.

Descripció de les funcions

menu_quantificacio_Callback

- Mostra el panel mida gra manual.
- Activa l'eix 2 i hi mostra la imatge original
- Activa l'eix 1 i hi mostra la imatge original
- Dibuixa l'escala

quantificacio_slider_brillantor_Callback

- Llegeix el valor del cursor.
- Suma o resta aquest valor a tota la imatge.

quantificacio_pushbutton_grisos_Callback

- Converteix la imatge color a tons de gris.

quantificacio_pushbutton_contrast_Callback

- Obre una finestra volada amb l'eina de MATLAB d'ajust de la imatge.

quantificacio_pushbutton_imatge_original_Callback

- Recupera la imatge original en els dos eixos.

quantificacio_slider_binarització_Callback

- Converteix la imatge a b/n a partir del llindar de binarització del cursor i la mostra per pantalla.
- Efectua una operació morfològica d'obertura, per tapar forats.
- Si està actiu el semàfor de dividir pantalla, genera una imatge partida a partir de la imatge binaritzada i l'original, i la mostra per pantalla.
- Calcula els percentatges de fase clara i fase fosca i els mostra al panel.

quantificacio_pushbutton_dividir_pantalla_Callback

- Activa/desactiva el semàfor de devidir_pantalla.
- Si és la primera vegada, genera una imatge partida a partir de la imatge binaritzada i l'original, i la mostra per pantalla.

quantificacio_pushbutton_automatic_Callback

- Converteix la imatge color a tons de gris.
- Executa la rutina de compensació automàtica de la il·luminació.
- Converteix la imatge a b/n a partir del llindar de binarització del cursor i la mostra per pantalla.
- Calcula els percentatges de fase clara i fase fosca i els mostra al panel.

Compensacio_automatica_illuminacio

- Divideix la imatge en 12 franges verticals.
- Suma el valor de tots els punts de cada franja.
- Calcula les diferències entre el valor de cada franja i la primera.
- Calcula el valor a sumar o restar a cada punt de cada franja, de manera que la suma de tots els valors de cada franja sigui igual a la primera.
- Construeix un vector de compensació horitzontal amb els valors anteriors, interpolant la resta.
- Genera una matriu de compensació replicant el vector de compensació tantes vegades com files té la imatge.
- Resta la matriu de compensació de la imatge original.
- Repeteix, de nou, tot el procés anterior, canviant files per columnes i a l'inrevés.

6.7. Compensar la il·luminació del microscopi a partir d'una imatge patró.

Si el focus d'il·luminació del microscopi no està ben centrat i la llum no es reparteix de manera uniforme per tot el camp d'observació, aleshores les imatges fotografiades presentaran uns gradients de brillantor, o de nivell que, si bé no entorpeixen massa la seva visió, afecten molt a les rutines automàtiques de càlcul.

En aquests casos es pot intentar millorar el problema fent un tractament matemàtic de la imatge.

En el capítol anterior s'ha explicat un procediment per compensar una il·luminació mal repartida. El problema d'aquell procediment és que es basa en que les fases de la mostra estiguin repartides de manera homogènia per tota la mostra. Si el cas és diferent, el resultat d'aquell procediment no serà bo.

Un altre procediment que se sol utilitzar consisteix en compensar la imatge mitjançant una imatge patró de la il·luminació.

La base teòrica és senzilla: si un punt de la mostra rep més intensitat lluminosa que un altre, aquests punts sempre rebran la mateixa intensitat, tant si hi ha mostra com si no (al menys durant un cert temps); per tant, si es fa una fotografia del camp d'observació del microscopi sense cap mostra, aquesta fotografia conté els valors de la intensitat lluminosa de cada punt del camp d'observació. Si es divideix una imatge presa amb aquella il·luminació per la imatge de la il·luminació (sense cap mostra) el resultat serà independent de la intensitat lluminosa.

Compensació de la il·luminació mitjançant una imatge patró

Aquest procediment implica que, a l'hora de prendre les fotos de les mostres, també es prengui una foto del camp d'observació sense cap mostra. Aquesta serà la imatge patró. En la ***Figura 6.16*** es veuen la imatge d'una mostra i la imatge patró.

Tanmateix, si es treballa amb imatges en color (generalment RGB), sol succeir que la imatge patró està acolorida.

En la mateixa ***Figura 6.16*** es veu que la imatge patró és de color taronja. Això significa que els components R i G d'aquesta imatge patró seran força més elevats que no pas el component B.

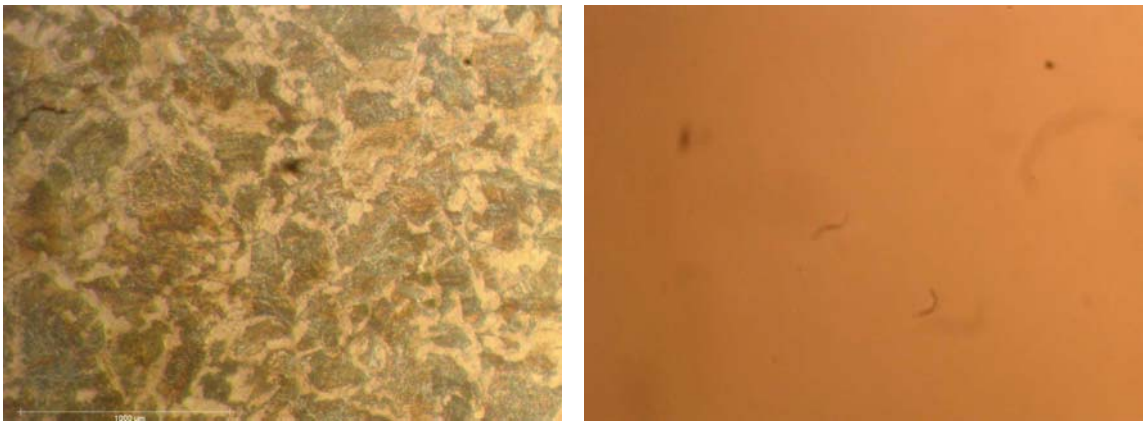


Figura 6.16*. Imatge d'una mostra i imatge patró per a compensar la il·luminació.*

En el moment de dividir la imatge original per la imatge patró, els component R i G de la imatge original es veuran molt més atenuats que no pas el component B. Això vol dir que la imatge final tindrà una clara predominança de blaus, com es pot veure en la imatge resultant:

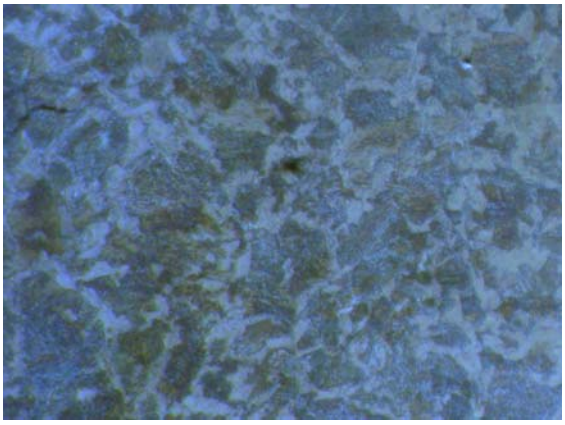


***Figura 6.17**. Imatge amb la il·luminació compensada, però acolorida degut al color de la imatge patró.*

Per resoldre aquest problema hi ha dues solucions:

1. Si només interessa compensar la il·luminació per a efectuar la quantificació de les fases d'una mostra, aleshores desapareix el problema, ja que en el procés de quantificació de les fases la imatge es converteix a tons de gris, i desapareix el color.

2. En canvi, si interessa mantenir el color per efectuar altres mesures, o bé per observar la mostra final, aleshores caldrà fer un altre tractament a la imatge i recuperar el color original.

   En el programa s'ha implementat una rutina que ajusta els nivells de color de les imatges, com es pot veure en la ***Figura 6.18***. Aquesta rutina ajusta el nivell de color, el contrast i la brillantor general de la imatge.

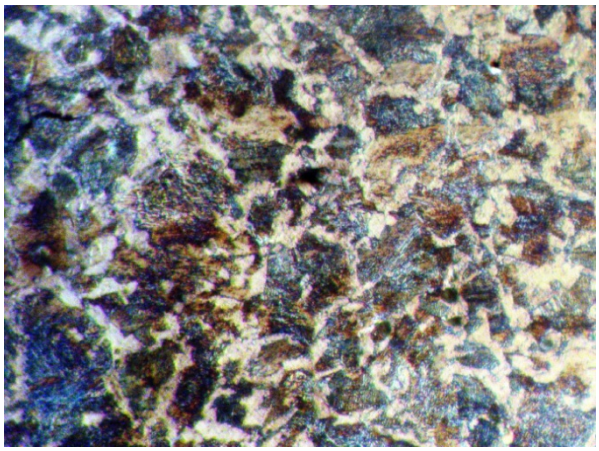


***Figura 6.18**. Efecte de l'ajust del color en la imatge anterior.*

Com a curiositat, es pot veure que la imatge de la ***Figura 6.18*** presenta uns colors diferents als de la imatge original de la ***Figura 6.16***. Això és degut a que la imatge original no té els colors anivellats (el resultat de fotografiar una mostra amb el color de llum taronja que es veu en la mateixa figura provoca que la imatge resultant també tingui excés de taronja). Així doncs, els colors originals de la mostra són els que es veuen en la ***Figura 6.18***, i no pas els de la ***Figura 6.16***.

Procediment de mesura

El procediment de compensació de la il·luminació del microscopi mitjançant una imatge patró consisteix en els passos següents:

1. Carregar la imatge que es vol compensar.

2. Activar el procediment de compensació a partir de

   Menu → Compensació il·luminació → Mitjançant imatge patró

3. Prémer el botó ***Carregar imatge patró***, i escollir la imatge patró en la finestra volada de càrrega. Una còpia petita de la imatge patró apareixerà a la part inferior del panel.

4. Prémer el botó ***Compensar la il·luminació***. S'aplicarà la rutina de compensació i el resultat apareixerà a la finestra principal.

5. En cas que es vulgui anivellar els color, prémer ***Nivells automàtics***. La rutina efectua un anivellament de colors, nivell i contrast, al mateix temps.

6. En cas que es vulgui ajustar el nivell de brillantor, es pot actuar sobre el cursor ***Ajust de la brillantor***.

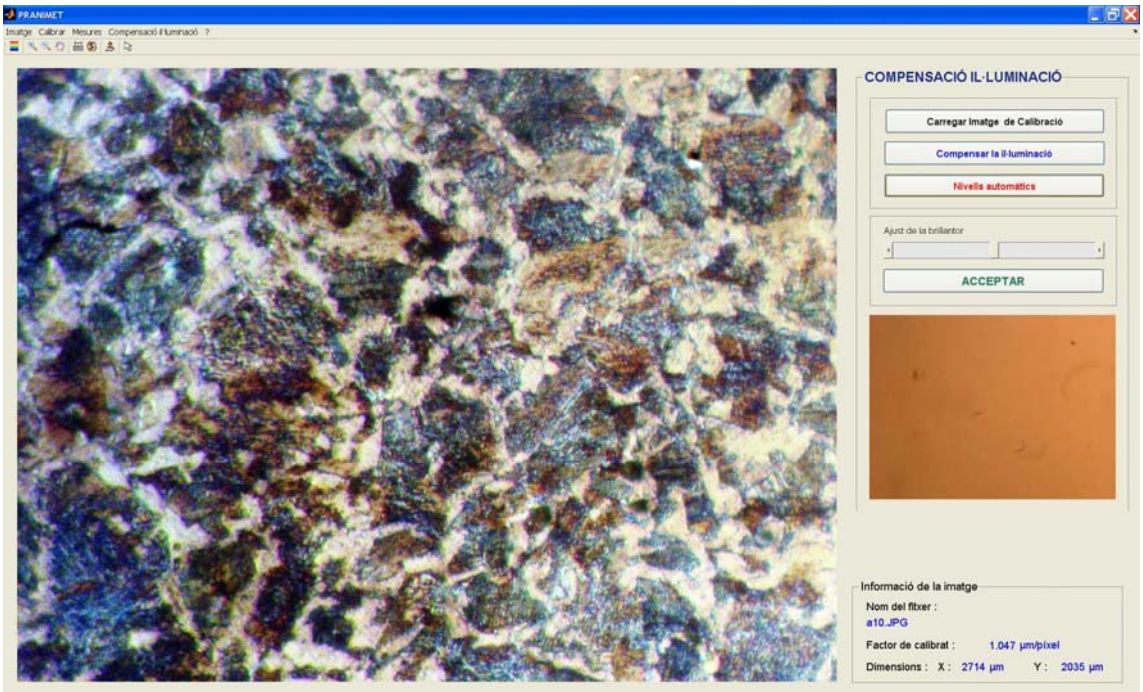


Figura 6.19*. Compensació de la il·luminació mitjançant una imatge patró.*

Descripció de les funcions

menu_compensacio_illuminacio_Callback

- Mostra el panel de compensació de la il·luminació
- Inicialitza el cursor i mostra la imatge original

compensacio_illuminacio_pushbutton_llegir_patro_Callback

- Llegeix imatge patró, i la mostra al final del panel.

compensacio_illuminacio_pushbutton_compensar_Callback

- Divideix entre sí les dues imatges, i mostra el resultat per la pantalla.

compensacio_illuminacio_pushbutton_nivells_Callback

- Executa la rutina d'anivellament de colors, nivells i contrast, i mostra el resultat per la pantalla.

compensacio_slider_brillantor_CreateFcn

- Modifica el nivell de brillantor de la imatge, afegint un offset determinat pel cursor

compensacio_illuminacio_pushbutton_acabar_Callback

- Tanca el panel

6.8. Aplicació de les diverses eines.

S'han dissenyat 7 eines que es poden aplicar en qualsevol moment i en qualsevol procés de mesura que s'estigui efectuant:

Eines de Zoom

- Zoom in
- Zoom out
- Pan

Eines de mesura

- Mesura de distàncies
- Mesura de superfícies

Altres eines

- Configuració de colors i gruixos de línies
- Còpia de la imatge de la pantalla al portapapers

Eines de Zoom. 

Les tres eines de zoom ja estan implementades per defecte en GUIDE. Només cal activar-les i col·locar-les a la barra d'eines, tal com s'ha fet.

Eines de Mesura de distàncies i de superfícies. 

Són unes eines fonamentals en el programa, ja que una de les especificacions era poder mesurar mides de grans, gruixos de capes, etc.

Per a l'eina de distàncies s'ha aprofitat la rutina que mesura la distància del micròmetre en el procés de calibrat, modificant la part de visualització de la mesura. Com que la mesura de distàncies es pot realitzar en qualsevol direcció dintre de la imatge, el seu resultat pot fer nosa si es col·loca sobre o sota de la línia, atès que la línia pot ser horitzontal, vertical o inclinada.

S'ha realitzat una rutina que calcula la inclinació de la línia dibuixada i situa el resultat de la mesura en funció d'aquesta inclinació. A continuació, i també com a exemple de la programació en MATLAB, s'inclou la funció ***mesura_linia*** àmpliament comentada.

```
% ********************************************************************
% ***** BOTÓ DE MESURAR DISTÀNCIES *************************************
% ********************************************************************

% La funció s'anomena mesura_linia_ClickedCallback
% La paraula ClickedCallback l'afegeix GUIDE, per indicar que es tracta
% d'una eina que està situada a la barra d'eines. S'activarà en el moment
% que es premi aquesta eina amb el ratolí.
% ------------------------------------------------------------------
function mesura_linia_ClickedCallback(hObject, eventdata, handles)
% Es defineixen les variables escala i op com a globals, de manera que la
% funció pugui anar a buscar els seus valors a altres funcions.
global escala;
global op;
% Força Axes1 com els eixos actius (imatge principal)
axes(handles.axes1);
% Intenta executar la funció externa dibuixar_escala que, evidentment,
% dibuixa l'escala sobre la imatge principal. En cas que no pugui (perquè
% encara no s'hagi calibrat la imatge) executa les instruccions que hi han
% després de catch me (com que no n'hi ha cap, no farà res).
try
   dibuixar_escala();
catch me
end
% És una funció de MATLAB que permet que l'usuari dibuixi una línia de
% manera interactiva. A la línia dibuixada li assigna un handle, que el
% desa a la variable hml. El color de la línia vindrà especificat per les
```

```
%  opcions del programa, desades en la matriu op.
hml=imline;
setColor(hml,op(5).c);
%  El programa espera que l'usuari validi la línia fent un doble click al
%  seu damunt, llegeix les seves coordenades, i les desa a la matriu linia.
coord = wait(hml);
linia=getPosition(hml);
%  Esborra l'objecte hml. Com que hml és el handle o apuntador a la línia
%  dibuixada, aleshores esborra la línia diguixada abans.
delete(hml);
%  Dibuixa una nova línia a sobre justament de l'anterior, amb totes les
%  característiques desades a la matriu d'opcions (color, gruix, estil de
%  línia, estil del marcador final, mida d'aquest marcador i nom de la
%  línia).
line([linia(1,1) linia(2,1)],[linia(1,2) linia(2,2)],...
   'Color',op(6).c,'LineWidth',op(6).w,'LineStyle',op(6).s,'Marker',op(6).m,...
   'MarkerSize',op(6).ms,'Tag','linia_mesura');
%  Calcula la projecció horitzontal (Dx) i vertical (Dy) de la línia.
Dx=linia(2,1)-linia(1,1);
Dy=linia(2,2)-linia(1,2);
%  Calcula l'angle que forma la línia respecte la horitzontal.
angle=(-1)*atan(Dy/Dx);
%  Calcula la longitud de la línia per Pitàgores. Ho multiplica pel factor
%  de calibrat per passar de píxels a micres.
long=((Dx*Dx+Dy*Dy)^0.5)*escala;
%  Calcula el punt central de la línia.
centrex=round((linia(1,1)+linia(2,1))/2);
centrey=round((linia(1,2)+linia(2,2))/2);
%  Aplica una funció experimental per trobar la posició inicial del text de
%  la mesura, a partir del centre de la línia i del seu angle
tx=150*sin(angle)-90;
ty=50*cos(angle);
%  Escriu el text amb les dimensions de la línia amb totes les
%  característiques desades a la matriu d'opcions (2 decimals, color, mida
%  de lletra i nom de l'objecte).
text(centrex+tx,centrey+ty,sprintf('%0.2f µm',long),...
   'Color',op(7).c,'FontSize',op(7).w,'Tag','text_mesura');
```

Per a l'eina de superfícies s'ha aprofitat la rutina que mesura la superfície dels grans, en la *mesura de la mida aparent del gra de manera manual*, adaptant-la i introduint el valor de la mesura realitzada.

En la ***Figura 6.20*** es mostra el resultat de mesures efectuades amb aquestes eines.

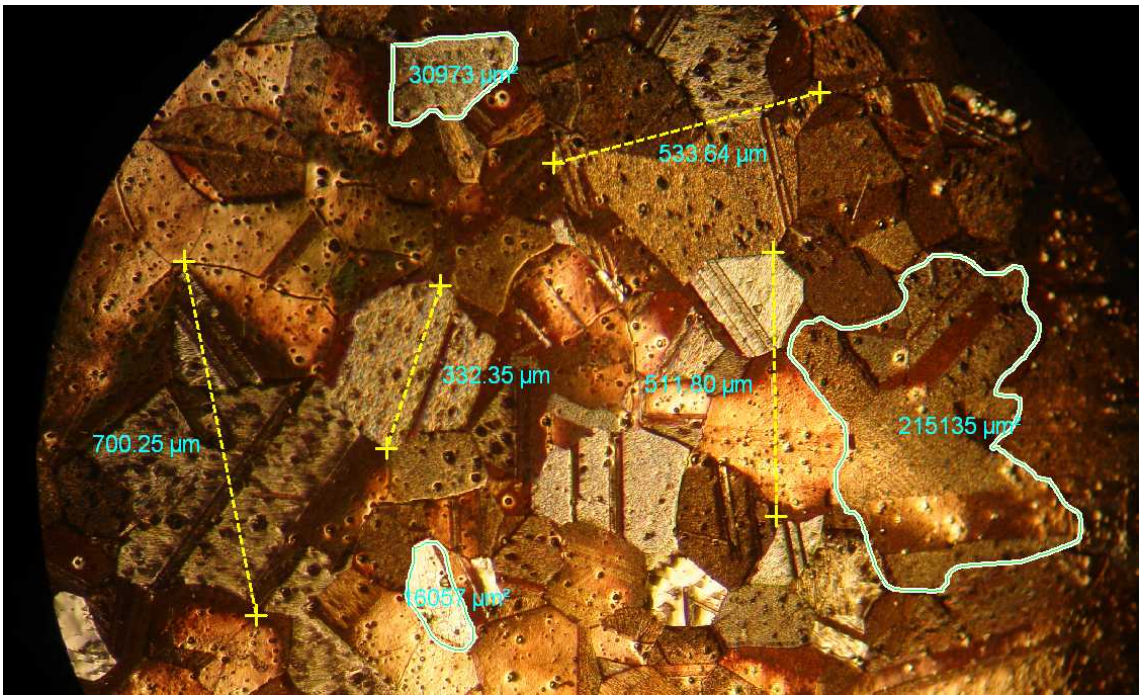


***Figura 6.20**. Mesures de distàncies i de superfícies.*

Eines de Configuració, Còpia al portapapers i Nivells Automàtics.

L'eina de ***Configuració*** permet canviar els paràmetres bàsics de les línies i els textos que es dibuixen sobre la imatge original.

De les línies es poden modificar els següents paràmetres:

- Color
- Gruix
- Estil de línia
- Estil de marcador
- Mida del marcador

Dels textos es poden modificar els següents paràmetres:

- Color
- Mida de la Font

Quan es prem la corresponent icona apareix el panel de configuració que permet modificar tots els paràmetres que s'han dit abans.

Si es prem el botó ***Tancar panel de configuració***, valida els canvis, tanca el panel i retorna al procés anterior.

Si es prem el botó ***Desar configuració per defecte***, valida els canvis, els desa en el fitxer *opcions.mat* (fitxer de configuració que llegeix el programa quan s'engega), tanca el panel i retorna al procés anterior.

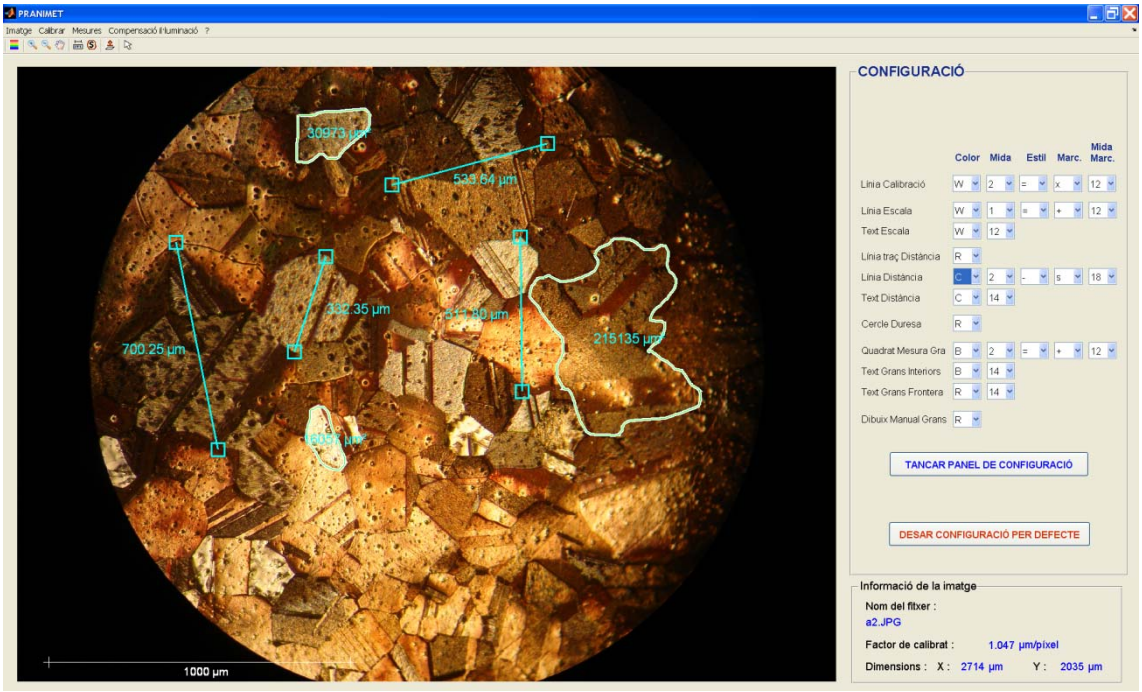


Figura 6.20*. Panel de configuració, i el seu efecte canviant les propietats de les línies.*

L'eina de ***Còpia al portapapers*** copia la finestra activa (tot el que es veu per la pantalla) al portapapers, de manera que es pot enganxar en qualsevol altra aplicació (apareix un missatge d'avís a l'acabar).

L'eina de ***Nivells Automàtics*** realitza un ajust automàtic dels nivells de color, contrast i brillantor de la imatge activa. Funciona com un interruptor, de manera que es pot activar i desactivar.

7. Resolució i exactitud

Com ja s'ha comentat a la introducció, PRANIMET pertany a la família dels *instruments virtuals*. La diferència entre un instrument 'tradicional' i un instrument 'virtual' rau en el fet que els primers utilitzen els recursos electrònics tradicionals en la seva construcció, mentre que els segons solen utilitzar massivament els recursos informàtics. La idea general que es té d'un instrument virtual és la d'un ordinador de propòsit general en el que està funcionant un programa informàtic que actua d'una manera similar a un instrument electrònic tradicional.

Tanmateix, no es pot oblidar que, tant l'un com l'altre, són instruments i, per tant, com que serveixen per mesurar, cal indicar clarament les seves especificacions (resolució, exactitud, precisió, etc.).

En aquest sentit, PRANIMET és un instrument que serveix per mesurar uns determinats paràmetres de provetes metal·logràfiques. Cal, doncs, estudiar el seu comportament.

En aquest capítol es presenta un estudi de la resolució i exactitud de les mesures fetes amb PRANIMET.

7.1. Resolució

El procediment que utilitza PRANIMET per mesurar és diferent del que utilitzen la majoria d'instruments tradicionals. Les mesures es prenen, directament, sobre la imatge que s'està mesurant. És a dir, es mesuren (es compten) els píxels: si es mesura una distància, es compten els píxels que separen els seus extrems; si es mesura una superfície, es compten els píxels inclosos en el seu contorn.

Si s'entén que la resolució d'un instrument és la mínima variació de l'entrada que és capaç de detectar l'instrument, aleshores la resolució de PRANIMET està donada, directament, per la mida dels píxels, és a dir, per la resolució de la imatge. Com que la resolució de la imatge la fixa la càmera fotogràfica, aleshores la resolució del programa depèn de la resolució de la càmera fotogràfica que s'utilitzi per fer les fotografies de les provetes.

De tota manera, a l'hora de donar la resolució del programa interessa donar-la en funció de la dimensió que el programa està mesurant, és a dir, en micres, i no pas en píxels.

El procés de calibrat efectua, precisament, aquesta conversió: relaciona la longitud en píxels del micròmetre amb la seva longitud en micres.

Per exemple, en les imatges de prova que s'han utilitzat en la confecció del projecte, el factor de calibrat en l'escala x10 x1 val

$$fc = 1.047\ \mu m/píxel$$

Això significa que, en aquesta escala, cada píxel de la imatge mesura 1.047 µm (en qualsevol de les dues direccions X, Y).

Per tant, la resolució del programa en cada escala està donada, directament, pel corresponent factor de calibrat.

7.2. Exactitud

L'exactitud dóna el marge de confiança d'una mesura. Sempre és igual o pitjor que la resolució; mai pot ser millor.

L'exactitud depèn de molts factors, com la resolució, la construcció de l'equip, les condicions ambientals, les condicions d'observació, etc. Tots aquests factors condicionen que la mesura no es faci sempre en unes condicions òptimes, ni amb uns instruments òptims, de manera que el resultat final tindrà una certa incertesa. Quan més bo sigui l'instrument i les condicions en què es fa la mesura, més exacta serà.

L'exactitud es pot donar de dues maneres diferents:

1. Proporcionant la fita de l'error absolut al efectuar una mesura.
2. Proporcionant la fita de l'error relatiu al efectuar la mesura.

L'error absolut es defineix com la diferència entre el valor real que s'està mesurant i el valor obtingut en la mesura:

$$e_a = v_r - v_m \tag{7.1}$$

L'error relatiu es defineix com l'error absolut dividit pel valor real:

$$e_r = \frac{e_a}{v_r} = \frac{v_r - v_m}{v_r} \cong \frac{v_r - v_m}{v_m} \tag{7.2}$$

En l'apartat anterior s'ha establert que la resolució del programa depèn, directament, de la resolució de la imatge. Atès que un dels factors més importants per a calcular l'exactitud és, precisament, la resolució de l'instrument, hom es pot demanar si un increment de resolució comportarà sempre una millora de l'exactitud.

La resposta no és afirmativa: no s'ha de suposar que incrementant la resolució de la càmera fins a alts nivells, l'exactitud del programa també ho farà: un increment important de la resolució de la càmera comportarà un decrement important de les dimensions dels píxels. Com que la situació dels punts inicial i final d'una mesura els col·loca l'observador, a mà, sobre la imatge, arribarà un moment en què es presentarà una indeterminació important a l'hora de situar el punt inicial o final de la mesura. A partir d'aquest moment, un increment de la resolució de la imatge no comportarà pas cap increment de l'exactitud de la mesura.

La ***Figura 7.1*** mostra una imatge del programa en el moment d'ajustar el cercle damunt d'una empremta Vickers, per fer una mesura de duresa:

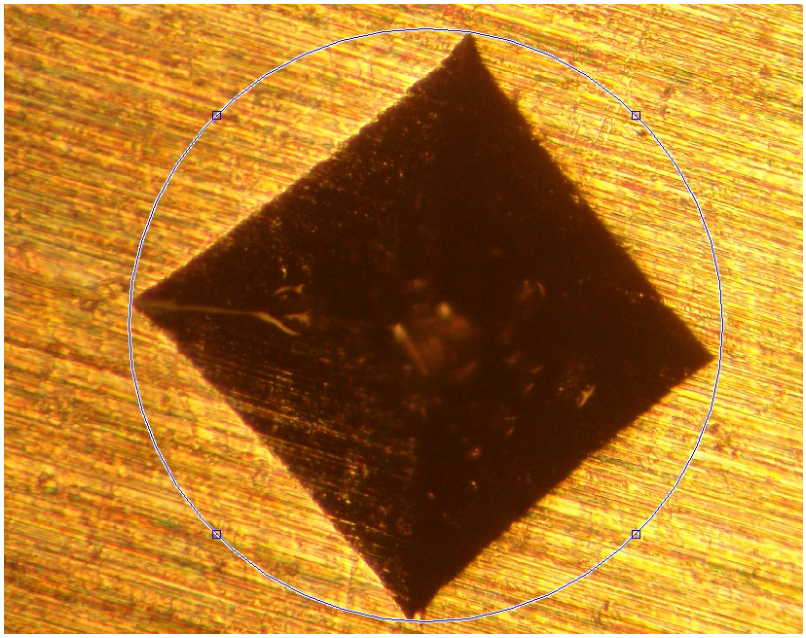


Figura 7.1. *Ajust del cercle en una mesura de duresa Vickers.*

Per ajustar més bé el cercle, i no perdre exactitud, es fa un zoom de la part superior de l'empremta. En la ***Figura 7.2*** es mostra el zoom efectuat, de manera que es poden distingir els píxels, i permet ajustar exactament el cercle.

Les qüestions que sorgeixen al veure aquesta imatge són les següents:

Està ben col·locat el cercle, o ha d'anar més amunt, o més avall?

On comença i on acaba l'empremta Vickers?

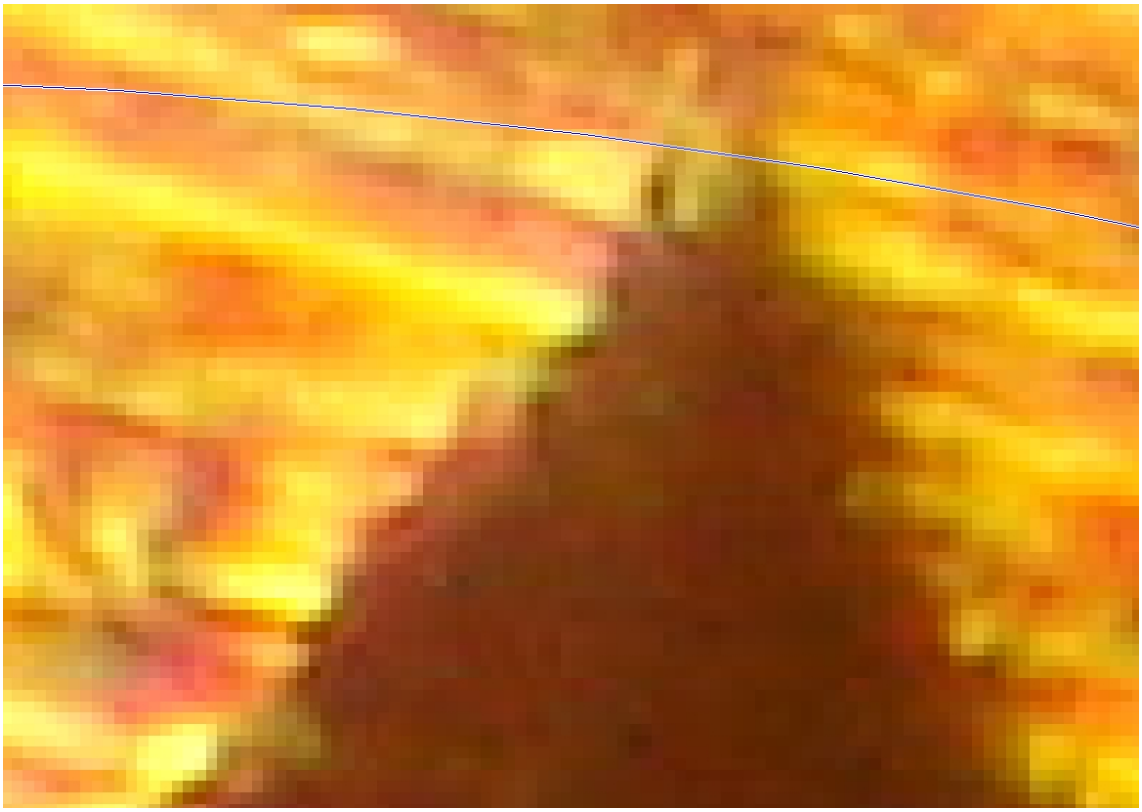


Figura 7.2*. Zoom de la imatge anterior on es veu la indeterminació a l'hora d'ajustar el cercle a l'empremta Vickers.*

La imatge està presa amb una càmera de 5.1 Mpíxels, i té unes dimensions de 2592x1944 píxels.

A la vista de la imatge anterior, està clar que un increment de resolució de la imatge comportarà un increment de resolució del programa, però no comportarà pas un increment d'exactitud de la mesura.

En definitiva, l'exactitud final de la mesura dependrà de molts factors, alguns dels quals són:

- La resolució de la imatge.
- La qualitat de la càmera fotogràfica
- La resolució del microscopi (el poder de separació)
- El nivell de zoom utilitzat.
- La qualitat de la mostra.
- L'habilitat i l'agudesa visual de l'observador, etc.

A continuació s'inclou un procediment per a calcular l'exactitud de les mesures efectuades amb el programa PRANIMET.

L'exactitud d'una mesura depèn de dos factors:

1. L'exactitud del procés de calibrat.
2. L'exactitud de cada procés de mesura.

El procés de calibrat és bàsic, atès que s'hi calcula el factor de calibrat, que servirà per a calcular totes les altres mesures. L'exactitud del procés de calibrat influeix directament sobre l'exactitud de les altres mesures.

Exactitud del procés de calibrat:

Es pot definir la fita de l'error produït en el procés de calibrat com la suma de tots els errors produïts en el procés. En aquest cas es consideren, només, els més importants :

$$e_{PC} = \pm(e_t + e_i + e_c) \quad [píxels] \tag{7.3}$$

on e_{PC} és l'error del procés de calibrat
e_t és l'error degut a la imprecisió en el traç de la línia
e_i és l'error degut a la inclinació del micròmetre
e_c és l'error degut als càlculs

Error degut a la imprecisió en el traç de la línia:

Atès que el procés de calibrat compta els píxels entre dos punts del micròmetre, directament sobre la pantalla de l'ordinador, si la visualització és dolenta, o la imatge és petita, el resultat serà molt imprecís. Per a obtenir el màxim de precisió, en el disseny del programa s'han introduït les solucions següents:

1. Fer que la imatge sigui el més gran possible.
2. Implementar una eina de Zoom que permeti ampliar la imatge fins al nivell dels píxels.

D'aquesta manera, el procés de situar els punts inicial i final de la línia que s'ha de calibrar es pot fer amb la màxima precisió possible.

En tots els càlculs d'errors que es realitzen en aquest capítol se suposa que el traçat de les línies i la col·locació dels altres elements gràfics s'efectua amb la màxima precisió possible, utilitzant l'eina zoom, si convé.

Per tant, ha de quedar clar que si es vol mantenir els errors dins dels marges obtinguts en aquest apartat, caldrà efectuar les mesures de manera correcta, ampliant la imatge si cal.

Atès que es pot ampliar fins a l'últim detall, es considera que l'error en la col·locació del punt inicial i del punt final de la línia és de ±1 píxel. En la ***Figura 7.3*** es mostra com, gràcies a la utilització del zoom, és possible situar l'origen de la línia justament a sobre del píxel corresponent.

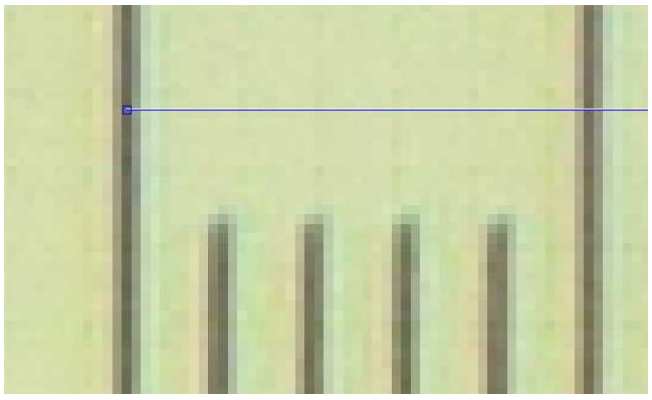


Figura 7.3*. Error degut a la imprecisió al dibuixar la línia.*

Com que el dibuix de la línia implica situar dos punts, el punt inicial i el punt final, aleshores l'error degut a la imprecisió en el traç de la línia serà:

$$e_t = 2 \times \pm 1\ píxel = \pm 2\ píxels \tag{7.4}$$

Error degut a la inclinació del micròmetre:

Quan es calibra la imatge, el programa traça una línia perfectament horitzontal. En cas que el micròmetre estigui una mica inclinat, la longitud mesurada serà superior o inferior a la real, en funció de com es faci la mesura.

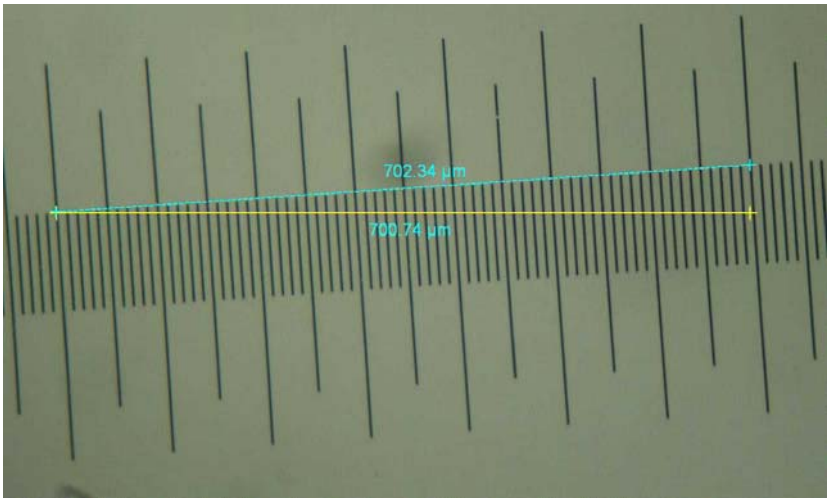


***Figura 7.4**. Error degut a la inclinació del micròmetre.*

Tal com es veu en la ***Figura 7.4***, la diferència màxima serà la projecció de la mesura feta, o de la línia mesurada, i serà funció de la longitud mesurada i de l'angle d'inclinació del micròmetre.

Suposant que l'angle d'inclinació sigui α, que la longitud mesurada sigui l_m i que la longitud real sigui l_r, es complirà que:

$$l_m = l_r \times Cos(\alpha) \tag{7.5}$$

L'error absolut degut a la inclinació del micròmetre serà:

$$e_i = l_r - l_m = \frac{l_m}{Cos(\alpha)} - l_m \tag{7.6}$$

$$e_i = \pm\left[l_m \times \left(\frac{1}{\cos(\alpha)} - 1\right)\right] \ [píxels] \qquad (7.7)$$

Aquest error es pot minimitzar mesurant de manera rigorosa i repetint la fotografia del micròmetre en cas que hagi quedat massa inclinat. En la ***Figura 7.4*** la inclinació és molt evident, i es tracta tan sols de 3°.

Error degut als càlculs:

Per una banda, els càlculs es realitzen sense cap tipus d'arrodoniment o de simplificació, fins al final.

Per l'altra banda, MATLAB realitza totes les operacions amb nombres de doble precisió i coma flotant. Això significa que cada nombre està format per 15 o 16 xifres significatives, entre mantissa i exponent.

A partir d'aquestes dues premisses es dedueix que els errors de càlcul (errors d'arrodoniment i de simplificació) són pràcticament zero.

Error final del procés de calibrat:

A partir de les equacions anteriors, la fita de l'error del procés de calibrat és:

$$e_{PC} = e_t + e_i + e_c = \ \pm 2 \ \pm\left[l_m \times \left(\frac{1}{\cos(\alpha)} - 1\right)\right] + 0 \quad [píxels] \qquad (7.8)$$

Si es vol expressar aquest error en micres, s'ha de multiplicar per la sensibilitat de l'instrument, és a dir, pel factor de calibrat.

El factor de calibrat es defineix com:

$$factor\ de\ calibrat = fc = \frac{núm.\ de\ micres\ del\ micròmetre}{longitud\ en\ píxels\ del\ micròmetre} = \frac{l_r}{l_m} \qquad (7.9)$$

Per tant,

$$e_{PC} = \pm \left[2 + l_m \times \left(\frac{1}{\cos(\alpha)} - 1\right)\right] \times \frac{l_r}{l_m} \qquad [\mu m] \qquad (7.10)$$

I, en definitiva,

$$e_{PC} = \pm \left[2fc + l_r \times \left(\frac{1}{\cos(\alpha)} - 1\right)\right] \qquad [\mu m] \qquad (7.11)$$

Suposant un error màxim de ±1 píxel en la situació dels punts extrems de la recta, i una inclinació màxima de 2° del micròmetre, l'exactitud del procés de calibrat per a cada una de les possibles combinacions microscopi-càmera és la següent:

***Taula 7.1**. Error absolut del procés de calibrat.*

Escala microscopi	Zoom Càmera	l_m(µm)	Fc (µm/píxel)	Exactitud
x 2.5	x 1	1000	4.2459	±9.2µm
x 2.5	x 3.1	1000	1.4863	±3.6µm
x 10	x 1	1000	1.0470	±2.8µm
x 10	x 3.1	900	0.37989	±1.4µm
x 50	x 1	430	0.21026	±0.69µm
x 50	x 3.1	180	0.073423	±0.26µm

Aquests valors són experimentals, i s'han obtingut amb el programa PRANIMET, a partir d'imatges reals obtingudes amb els instruments del Laboratori de Mecànica de la UVic.

A partir dels errors trobats en el procés de calibrat, es pot expressar correctament el factor de calibrat en cada escala.

A continuació s'efectua el càlcul per a una escala. Per a totes les demés, se segueix el mateix procediment:

Escala x2.5 x1

$$error\ al\ mesurar\ el\ micròmetre = \frac{e_{PC}}{l_r} = \frac{9.2\mu m}{1000\mu m} = \pm 0.0092 \qquad (7.12)$$

$$error\ del\ factor\ de\ calibrat = e_{fc} = fc \times error\ mesura \qquad (7.13)$$

$$e_{fc} = 4.2459\ \mu m/píxel \times 0.0092 = \pm 0.03906\ \mu m/píxel \qquad (7.14)$$

Finalment, el factor de calibrat en aquesta escala val:

$$fc = (4.2459\ \pm 0.03906)\mu m \cong (4.25\ \pm 0.04)\ \mu m/píxel \qquad (7.15)$$

Procedint de la mateixa manera per a cada escala, s'obtenen els valors correctes dels diferents factors de calibrat, que es mostren en la ***Taula 7.2.***

Taula 7.2*. Valors del factor de calibrat, tenint en compte l'error del procés.*

Escala microscopi	Zoom Càmera	Factor de calibrat	Error relatiu
x 2.5	x 1	$(4.25\ \pm 0.04)\ \mu m/píxel$	±0.94%
x 2.5	x 3.1	$(1.486\ \pm 0.006)\ \mu m/píxel$	±0.40%
x 10	x 1	$(1.047\ \pm 0.003)\ \mu m/píxel$	±0.29%
x 10	x 3.1	$(0.3799\ \pm 0.0006)\ \mu m/píxel$	±0.16%
x 50	x 1	$(0.2126\ \pm 0.0004)\ \mu m/píxel$	±0.19%
x 50	x 3.1	$(0.0734\ \pm 0.0002)\ \mu m/píxel$	±0.27%

El motiu de la diferència que hi ha entre els diferents errors es deu, senzillament, als processos gràfics de la mesura:

En l'escala x2.5 x1, l'error absolut al mesurar el micròmetre és de ±9.2µm, però el micròmetre ocupa molts pocs píxels de la imatge. Per tant, l'error relatiu serà molt gran. En canvi, en l'escala x10 x3.1, el micròmetre pràcticament ocupa tota la imatge; per tant, l'error relatiu serà més petit.

Aquest resultat corrobora una de les normes més elementals de la instrumentació: sempre cal mesurar en l'escala adequada, de manera que la mesura feta s'acosti el màxim possible al fons d'escala (ocupi el màxim espai de l'instrument); d'aquesta manera l'error provocat en la mesura serà mínim.

Es pot veure que tot i tractar-se d'un sistema de mesura manual i gràfic, l'error del procés de calibrat és força petit, entre un 0.16% i un 0.94%. En cas que es disposés d'un micròmetre de més longitud, de manera que ocupés tota la imatge, l'error de calibrat en totes les escales estaria al voltant del ±0.15%.

L'error d'indeterminació al situar els extrems de la línia es pot minimitzar utilitzant un micròmetre amb les línies verticals molt fines.

Exactitud dels diferents processos de mesura:

Hi ha tres tipus diferents de processos de mesura. Per una banda, aquells que mesuren directament una longitud sobre la imatge (mesura de duresa, utilització de l'eina de mesura de longituds,...). Per l'altra, els que mesuren superfícies (diàmetre aparent de gra, índex de gra, utilització de l'eina de mesura de superfícies,...). Finalment, aquells que utilitzen altres tècniques, com el comptatge de grans, etc.

a. Exactitud dels processos que mesuren longituds

Per mesurar una longitud només fa falta indicar al programa els dos punts extrems. L'error en la mesura vindrà donada bàsicament per la indeterminació a l'hora de situar aquests dos punts extrems sobre la imatge.

En el procediment que es desenrotlla en aquest apartat se suposa que aquesta indeterminació és mínima, i equival a ±1 píxel. En cas que s'acordi utilitzar un altre valor per la indeterminació només caldrà refer els càlculs amb el nou valor.

A diferència del procés de calibrat, en el que la línia sempre era horitzontal, en totes les altres mesures que fa el programa la direcció pot ser qualsevol.

En aquests casos, doncs, s'ha de modificar lleugerament el procés de càlcul dels errors.

En la ***Figura 7.5*** es pot veure l'error produït per una indeterminació de ±1 píxel. En les línies de mesura horitzontals i verticals, l'error màxim provocat és, precisament, de ±1 píxel.

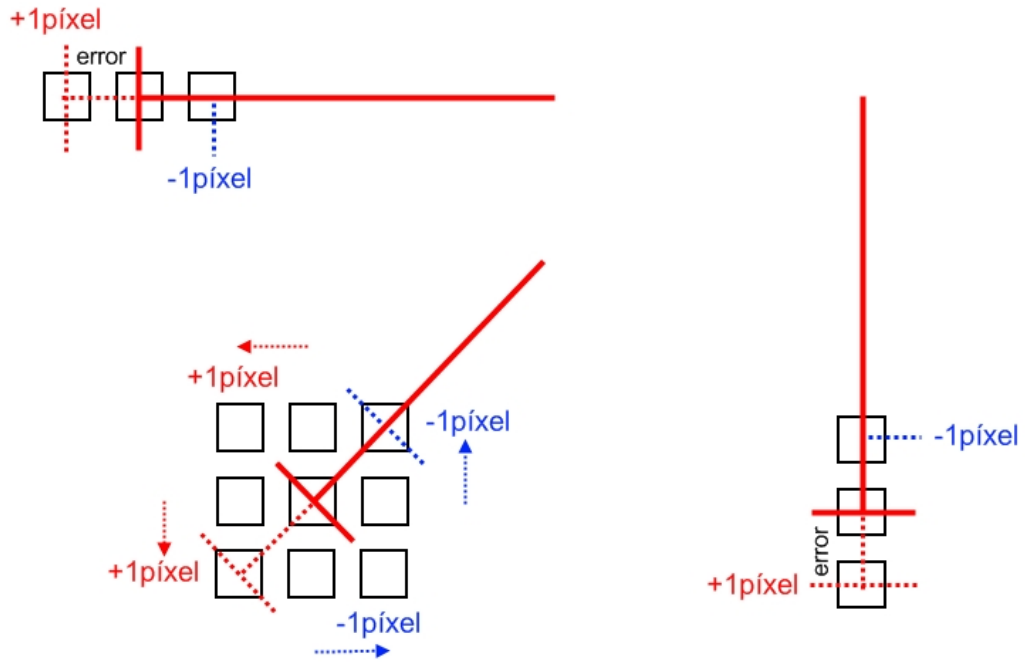


Figura 7.5*. Indeterminació a l'hora de situar els extrems de la línia.*

En canvi, quan la línia està inclinada, una indeterminació de ±1 píxel en la direcció X i en la direcció Y provocarà un error més gran, que es pot calcular per Pitàgores en el seu punt màxim (o mínim):

$$error = \sqrt{1^2 + 1^2} = \sqrt{2} = \pm 1.4142\ píxels \tag{7.16}$$

Com que aquest error pot aparèixer en els dos extrems de la línia, el màxim error comès, suposant una indeterminació de ±1 píxel a l'hora de situar els extrems de la línia, és:

$$e_l = 2 \times 1.4142 = \pm 2.8284\ píxels \cong \pm 3\ píxels \tag{7.17}$$

En la ***Taula 7.3*** es mostra el resum de l'exactitud del programa en cada escala, en el cas de mesura de longituds:

***Taula 7.3**. Recull dels errors del programa, suposant una indeterminació en totes les mesures de ±1 píxel en les direccions X i Y.*

Escala Microsc.	Zoom Càmera	Factor de calibrat	Error de mesura	Error de mesura
x 2.5	x 1	$(4.25 \pm 0.04)\ \mu m/píxel$	±3 píxels	±13 µm
x 2.5	x 3.1	$(1.486 \pm 0.006)\ \mu m/píxel$	±3 píxels	±4.2 µm
x 10	x 1	$(1.047 \pm 0.003)\ \mu m/píxel$	±3 píxels	±3.0 µm
x 10	x 3.1	$(0.3799 \pm 0.0006)\ \mu m/píxel$	±3 píxels	±1.1 µm
x 50	x 1	$(0.2126 \pm 0.0004)\ \mu m/píxel$	±3 píxels	±0.61 µm
x 50	x 3.1	$(0.0734 \pm 0.0002)\ \mu m/píxel$	±3 píxels	±0.21 µm

Exemple 1 de càlcul d'errors en longituds

Com a exercici d'aplicació d'aquests resultats, es calcularà la mesura correcta del diàmetre d'una empremta Vickers.

En la ***Figura 7.6*** es mostra el programa en el moment que s'ajusta el cercle a l'empremta Vickers.

Abans de fer la mesura s'ha hagut de calibrar el sistema. La fotografia de l'empremta Vickers està feta en l'escala x10 del microscopi, i amb un zoom de x3.1 de la càmera digital. Per tant, el sistema s'ha calibrat amb una fotografia del micròmetre feta en les mateixes condicions.

El programa indica que el diàmetre de l'empremta (o el que és el mateix, la diagonal) és de 494.88µm.

Segons la ***Taula 7.3.***, en l'escala x10 x3.1, el factor de calibrat és (0.3799 ±0.0006) µm/píxel, i l'error de la mesura és ±3 píxels o ±1.1µm.

La mesura feta és D = 494.88 µm

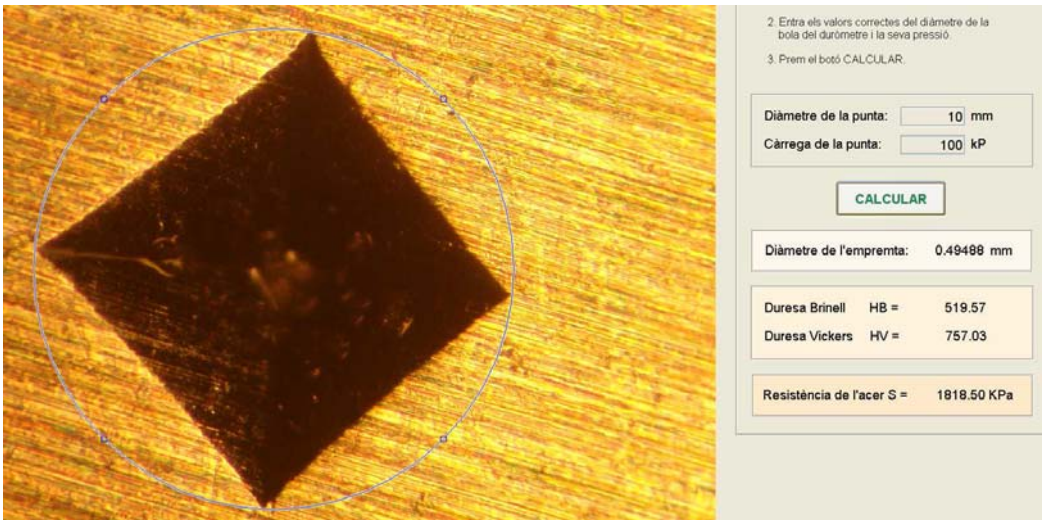


***Figura 7.6**. Càlcul de l'error en la mesura del diàmetre d'una empremta Vickers.*

L'error és

$$e_D = \pm(494.88 \text{ x } 0.0006 + 1.1\ \mu m)$$

$$e_D = \pm(0.2969 + 1.1) = \pm 1.3969 \approx \pm 1.4\ \mu m$$

Per tant, la mesura correcta del diàmetre serà:

$$D = (494.88 \pm 1.4)\ \mu m \approx (494.9 \pm 1.4)\ \mu m$$

Aquest procediment es pot aplicar a totes les altres mesures de longituds fetes amb el programa.

Tanmateix, i tal com passa amb tots els sistemes gràfics de mesura, l'error provocat pel programa pot passar a un segon terme si, per exemple, l'empremta del duròmetre no està ben definida. Quan passa això, l'usuari no té una marca o una referència clara a l'hora d'ajustar el cercle sobre l'empremta. Per cada píxel de diferència entre el diàmetre real de l'empremta i el del cercle, l'error s'incrementa o redueix, aproximadament, en $\frac{e_{PC}}{2}$. És a dir, en el cas de l'exemple anterior, cada píxel d'error que es produeixi al situar el cercle damunt de l'empremta ocasionarà un error de ±0.7 µm (si l'error es considera en els dos extrems, el valor serà ±1.4µm).

Malgrat això, si es treballa amb l'eina zoom i amb una mica de cura, el valor final de l'error continuarà essent petit.

Exemple 2 de càlcul d'errors en longituds

En aquest segon exemple se suposa que un procés de mesura d'una longitud en una imatge obtinguda en l'escala x10 del microscopi, i sense zoom de la càmera, ha proporcionat el resultat de 986.45µm.

Consultant la ***Taula 7.3*** es troba que en les condicions exposades en l'enunciat, el factor de calibrat és (1.047 ±0.003) µm/píxel. L'error al dibuixar la línia és de ±3 píxels o ±3 µm.

L'error és:

$$e_l = \pm(986.45 \times 0.003 + 3\ \mu m)$$

$$e_D = \pm(2.959 + 3) = \pm 5.959\ \mu m$$

Per tant, la mesura correcta de la longitud serà:

$$L = (986.5 \pm 5.959)\ \mu m \approx (987 \pm 6)\ \mu m$$

b. Exactitud dels processos que mesuren superfícies

Hi ha una diferència important entre la manera com PRANIMET mesura longituds i com mesura superfícies. Per mesurar una longitud només són necessaris dos punts, i per calcular l'error només fa falta saber la indeterminació que es pot produir al situar aquests dos punts sobre una imatge.

En canvi, per mesurar una superfície cal comptar tots els píxels que estan inclosos en un contorn determinat. Per calcular l'error caldria saber la indeterminació a l'hora de dibuixar cada un dels punts del contorn de la superfície que es vol mesurar. Atès que el dibuix del contorn es realitza de manera manual, amb el ratolí, és totalment impossible saber l'error efectuat al dibuixar el contorn.

En la ***Figura 7.7*** es pot veure (una mica exagerat) la indeterminació produïda al dibuixar el contorn d'un objecte:

***Figura 7.7**. Indeterminació a l'hora de dibuixar el contorn d'un objecte.*

Per tant, en aquesta memòria no es calcularà cap fita de l'error produït al mesurar superfícies.

Només a títol indicatiu, i amb la finalitat de tenir una idea de l'ordre de magnitud de l'error, es proposa un procediment aproximat de càlcul de l'error produït al mesurar superfícies mitjançant el programa PRANIMET:

- A partir de la superfície d'un objecte calculada pel programa es pot trobar el seu radi aparent, a partir de l'equació següent (es converteix l'objecte en una circumferència, amb la mateixa superfície que l'objecte):

$$r = \sqrt{\frac{S}{\pi}} \tag{7.18}$$

- Suposant un error o una indeterminació de ±1 píxel a l'hora de situar el radi, l'error en el càlcul de la superfície serà:

$$e_S = \pi[(r+1)^2 - (r-1)^2] = 4\pi r = \pm 2\pi r = \pm \pi D \tag{7.19}$$

  Així doncs, per cada ±1 píxel d'error a l'hora de situar el radi, l'error de la superfície s'incrementa ±πD

Exemple de càlcul d'errors en superfícies

Calcular l'error aproximat al mesurar un gra que té una superfície aproximada de 31416 µm², suposant que la indederminació en el procés de dibuixar el contorn del gra s'ha mantingut inferior a ±5 píxels.

Aplicant (7.18), s'obté el radi aparent de l'objecte r = 100µm

Aplicant (7.19) s'obté l'error de la superfície, considerant un error màxim d'indeterminació de ±1 píxel e_S = ±629 µm

Si l'error d'indeterminació pot ser 5 vegades més gran, aleshores l'error també: e_S = 5 x [±629 µm] = 3145 µm ≈ ±3200 µm

Per tant, la mesura de la superfície seria:

$$S = (31416 \pm 3200)\mu m^2 \approx (31400 \pm 3200)\mu m^2$$

Es pot comprovar que l'error és molt gran, de l'ordre del ±10%. Si es vol minimitzar caldria tenir més traça a l'hora de dibuixar el contorn, o dissenyar un procediment automatitzat de dibuix del contorn de l'objecte que asseguri una indeterminació menor.

Ha de quedar clar, de totes maneres, que aquest càlcul només s'ha inclòs a la memòria per proporcionar a l'usuari l'ordre de magnitud de l'error efectuat al mesurar una superfície

c. *Exactitud del procés que mesura l'índex de mida del gra segons UNE-NE ISO 643*

Atès que el programa implementa un **sistema de mesura d'acord amb la norma UNE-NE ISO 643**, cal saber quin és l'error en aquestes mesures.

El procediment consisteix en dibuixar un quadrat de superfície coneguda damunt de la mostra, i comptar manualment el nombre de grans que estan continguts enterament dins del quadrat, i els que estan a mitges. Al final s'aplica una equació que fa una certa mitja i calcula el nombre de grans per mm² i l'índex de gra. Aquest procediment, tot i estar reconegut com a ISO és poc exacte, i la mateixa

documentació ISO menciona que no es pot pas aconseguir una exactitud millor que ±0.5 punts en la mesura de l'índex de gra.

En aquesta mesura, el programa només proporciona eines de suport per a fer més fàcil l'aplicació del procediment. Per tant, no afegeix ni elimina error al procés de càlcul contemplat en la norma ISO mencionada.

d. *Exactitud del procés que quantifica els percentatges de les fases cristal·lines en una mostra*

El programa permet quantificar les fases cristal·lines d'una mostra, i proporciona dos procediments per a fer-ho.

D'una banda, implementa un **procediment manual**, amb els mateixos passos que se segueixen actualment (transformació d'imatge color a b/n, ajust del contrast, ajust dels nivells, binarització i càlcul final).

Per l'altra banda implementa un **procediment que automatitza** tots els passos excepte l'ajust del llindar de la binarització.

Atès que la quantificació de les fases depèn directament del valor del llindar en el procés de binarització, i aquest valor el proporciona l'usuari en funció d'una comparació visual entre la imatge binaritzada i l'original, no té sentit calcular els errors del procediment.

En qualsevol cas, el programa proporciona les eines perquè el procés de binarització sigui el més fidel possible a la imatge original, i perquè el procés de comparació de les dues imatges (la binaritzada i l'original) que determinarà el valor òptim del llindar de binarització es realitzi en les millors condicions possibles per a l'observador.

8. Resultats, millores i conclusions

En aquest capítol es valoren els resultats aconseguits en aquest treball, es proposen algunes millores i ampliacions per a treballs posteriors, i es formulen les conclusions a què s'ha arribat una vegada s'ha desenvolupat tot el projecte.

8.1. Resultats aconseguits

A continuació es repeteix l'objectiu del present TFM, i els requeriments que havia de satisfer el programa desenvolupat:

> *El TFM es titula **Programa d'anàlisi d'imatges de provetes metal·logràfiques**, i el seu objectiu és explorar les possibilitats del programa matemàtic MATLAB de MathWorks i la seva eina Entorn de Disseny d'Interfície Gràfica d'Usuari (GUIDE), desenvolupant un programa d'anàlisi d'imatges de provetes metal·logràfiques que es pugui utilitzar per a realitzar les pràctiques de laboratori de l'assignatura de Tecnologia de Materials de la titulació de Grau en Enginyeria Electrònica, que s'imparteix a la Universitat de Vic. El programa final s'ha de compilar, de manera que pugui funcionar en qualsevol ordinador de la Universitat o de l'alumnat (amb uns mínims requeriments tècnics), sense la necessitat d'utilitzar cap altre tipus de programari comercial.*

Requeriments del programa:

1. Es parteix d'imatges de microscòpia òptica, realitzades al Laboratori de Mecànica de la UVic mitjançant un microscopi òptic *Leica 123* i una càmera de fotos digital *Sony Cibershot* adaptada al microscopi. Es disposarà de:
    - Imatges de recristal·lització del coure.
    - Imatges d'acers.
    - Imatges de foses.
2. En cada un d'aquests casos es pretén fer les següents accions i obtenir les següents mesures:

a. Calibrar imatges de microscòpia.
b. Mesurar empremtes de duresa i calcular la duresa dels materials.
c. Mesurar els gruixos de les capes i altres distàncies en les mostres.
d. Mesurar els diàmetres dels grans i calcular el seu diàmetre mig.
e. Determinar l'índex de mida de gra.
f. Quantificar diferents fases cristal·lines (grafit en una fosa, perlita en un acer, inclusions de cuprita en el coure, etc.)

Resultats

1. Desenvolupament del *Programa d'anàlisi d'imatges de provetes metal·logràfiques.*

S'ha desenvolupat un programa d'anàlisi d'imatges de provetes metal·logràfiques (PRANIMET) que s'utilitzarà per a realitzar les pràctiques de laboratori de l'assignatura de Tecnologia de Materials de la titulació de Grau en Enginyeria Electrònica, que s'imparteix a la Universitat de Vic.

2. Nivell de satisfacció dels requeriments proposats

a. *Calibrar imatges de microscòpia*

El programa permet calibrar qualsevol tipus d'imatge de microscòpia, a partir de la fotografia d'un micròmetre. El resultat del procés és l'obtenció del paràmetre *factor de calibrat*. Una vegada calculat, qualsevol mesura que es faci sobre la imatge (en píxels) es pot passar a micres multiplicant-la pel corresponent *factor de calibrat*.

En la memòria s'ha determinat l'error produït per aquest procés, i s'ha proposat un procediment per a calcular-lo a partir de les imatges dels micròmetres obtingudes en el laboratori. En línies generals l'error és prou petit (inferior a ±1%), i es pot fer independent de l'escala si es disposa d'un micròmetre de la suficient longitud (quedaria per sota de ±0.15%).

b. *Mesurar empremtes de duresa i calcular la duresa dels materials.*

El programa permet mesurar empremtes de duresa Brinell i Vickers, mesurant el diàmetre de l'empremta Brinell i les diagonals de l'empremta Vickers.

El procediment de mesura utilitza un cercle que s'ajusta manualment damunt de l'empremta.

El programa proporciona el diàmetre de l'empremta Brinell, o la diagonal de l'empremta Vickers, en funció del cas. També calcula la duresa HB i la duresa HV, i la resistència de l'acer. Es proporcionen tots aquests resultats al mateix temps, i és tasca de l'usuari escollir quin resultat és vàlid en funció del mètode que s'utilitzi.

En la memòria s'ha determinat l'error produït pel procés de mesurar el diàmetre de l'empremta, i s'ha proposat un procediment per a calcular-lo.

A partir d'aquí, l'usuari pot calcular l'error de la duresa si disposa dels errors del diàmetre de la bola i de la càrrega del duròmetre Brinell, o de la càrrega de la punta en el cas del duròmetre Vickers.

En el cas del duròmetre Brinell,

$$HB = \frac{2P}{\pi D\left[D - \sqrt{D^2 - d^2}\right]} \tag{8.1}$$

on *D és el diàmetre de la punta del duròmetre*
d és el diàmetre de l'empremta
P és la càrrega del duròmetre

Si se saben els següents errors:

e_D és l'error del diàmetre de la bola del duròmetre
e_d és l'error de la mesura del diàmetre de l'empremta calculat abans
e_P és l'error de la càrrega de la punta del duròmetre

Aleshores l'error resultant del càlcul de l'equació (8.1) es:

$$e_{HB} = \sqrt{\frac{\partial(HB)^2}{\partial D} \times (e_D)^2 + \frac{\partial(HB)^2}{\partial d} \times (e_d)^2 + \frac{\partial(HB)^2}{\partial P} \times (e_P)^2} \qquad (8.2)$$

Tanmateix, i tal com passa amb tots els sistemes gràfics de mesura, l'error provocat pel programa pot passar a un segon terme si l'empremta del duròmetre no està ben definida. Quan passa això, l'usuari no té una marca o una referència clara a l'hora d'ajustar el cercle sobre l'empremta. Malgrat tot, si es treballa amb l'eina zoom i amb una mica de cura, el valor final de l'error continuarà essent molt petit.

c. Mesurar els gruixos de les capes i altres distàncies en les mostres.

El programa permet mesurar qualsevol tipus de distància directament sobre la imatge, marcant els dos punts extrems de la distància que es vol mesurar. Com a resultat del procés, el programa proporciona el resultat de la mesura en micres. El programa permet l'ús de l'eina Zoom per a aconseguir una major exactitud.

Com en el cas anterior, en la memòria s'ha determinat l'error produït pel procés de mesurar una distància, i s'ha proposat un procediment per a calcular-lo.

Com a millora addicional, s'ha afegit al programa una eina manual per a mesurar superfícies.

El procediment consisteix en que l'usuari dibuixi una figura tancada al voltant de l'objecte del qual vol mesurar la superfície. Aquesta superfície tancada s'ha d'ajustar el màxim al contorn de l'objecte. Una vegada dibuixada la figura, el programa calcula la quantitat de píxels que queden inclosos dintre de la figura, multiplica aquest valor per la superfície d'un píxel, i mostra el resultat, en micres, al centre de la figura dibuixada.

Atès que es tracta d'un procediment manual, l'exactitud del procés dependrà de l'habilitat de l'usuari en ajustar perfectament la línia traçada al contorn de l'objecte que es mesura. En la memòria s'ha proposat un procediment per a calcular l'ordre de magnitud de l'error de la mesura de

superfície. Evidentment, quan més precís sigui el dibuix del contorn, més petit serà l'error.

d. Mesurar els diàmetres dels grans i calcular el seu diàmetre mig aparent.
e. Determinar l'índex de mida de gra.

El programa implementa dos tipus de mesures:

D'una banda, implementa **un sistema de mesura manual** que permet mesurar la superfície de cada gra, el seu diàmetre aparent, i el diàmetre mig de tots els grans de la mostra. A partir d'aquests valors calcula el nombre de grans per mm^2 i l'índex de mida de gra.

El procediment és molt similar al que s'ha comentat anteriorment per a mesurar superfícies: l'usuari ha de resseguir manualment el contorn de cada gra. A partir d'aquí, tot el procés és automàtic.

Tot el que s'ha explicat sobre l'exactitud del procediment de mesurar superfícies s'aplica també a aquest cas. Tanmateix, aquí hi ha un element a favor: com que al final el sistema fa la mitja de tots els grans (de totes les mesures), l'error final queda força disminuït.

Per l'altra banda, el programa implementa un **sistema de mesura d'acord amb la norma UNE-NE ISO 643**. Aquest procediment consisteix en dibuixar un quadrat de superfície coneguda damunt de la mostra, i comptar manualment el nombre de grans que estan continguts enterament dins del quadrat, i els que estan a mitges. Al final s'aplica una equació que fa una certa mitja i calcula el nombre de grans per mm^2 i l'índex de gra. Aquest procediment, tot i estar reconegut com a ISO és molt poc exacte, i la mateixa documentació ISO menciona que no es pot pas aconseguir una exactitud millor que 0.5 en la mesura de l'índex de gra.

El programa implementat proporciona totes les eines i fa tots els càlculs, per tal de facilitar la mesura feta seguint aquesta norma (dibuixa el quadrat, proporciona les eines per marcar/desmarcar els grans interiors i els grans frontera, i fa tots els càlculs).

Com ja s'ha comentat en el capítol anterior, en aquesta mesura el programa ni afegeix ni elimina error al procés de càlcul contemplat en la norma ISO mencionada.

f. Quantificar diferents fases cristal·lines (grafit en una fosa, perlita en un acer, inclusions de cuprita en el coure, etc.)

El programa permet quantificar les fases cristal·lines d'una mostra, i proporciona dos procediments per a fer-ho.

D'una banda, implementa un **procediment manual**, amb els mateixos passos que se segueixen actualment (transformació d'imatge color a b/n, ajust del contrast, ajust dels nivells, binarització i càlcul final).

Per l'altra banda implementa un **procediment que automatitza** tots els passos excepte l'ajust del llindar de la binarització.

En aquest últim procediment s'ha incorporat una rutina que compensa automàticament els efectes deguts a una mala distribució de la il·luminació del microscopi. En totes les proves efectuades la rutina ha demostrat un funcionament impecable, proporcionant resultats fiables en mostres tan mal il·luminades que semblava impossible. L'únic requisit és que la distribució dels grans en la mostra sigui homogènia. Si els grans estan repartits de manera molt poc homogènia, la rutina no funciona bé. De tota manera, aquest cas és altament improbable.

Atès que la quantificació de les fases depèn directament del valor del llindar en el procés de binarització, i aquest valor el proporciona l'usuari en funció d'una comparació visual entre la imatge binaritzada i l'original, no té sentit calcular els errors del procediment.

En qualsevol cas, el programa proporciona les eines perquè el procés de binarització sigui el més fidel possible a la imatge original, i perquè el procés de comparació de les dues imatges (la binaritzada i l'original) que determinarà el valor òptim del llindar de binarització es realitzi en les millors condicions possibles per a l'observador.

8.2. Millores i ampliacions

Malgrat que el programa final es presenti com una aplicació tancada, en realitat és un entorn gràfic que controla una sèrie de procediments de mesura. Mantenint un esquelet comú, els diferents procediments són independents entre sí. D'aquesta manera es poden afegir nous procediments o modificar o esborrar els actuals de manera molt senzilla.

Així doncs, el programa s'ha estructurat de manera que sigui fàcilment ampliable amb altres rutines de mesura, o amb l'automatització de les rutines existents.

En una segona fase imminent, el programa es documentarà de manera exhaustiva i es depurarà amb la finalitat de minimitzar les variables utilitzades i millorar la velocitat de funcionament. Amb aquestes millores es pretén aconseguir que l'escriptura i organització del programa sigui fàcilment intel·ligible per a altres persones que vulguin millorar-lo o ampliar-lo.

8.3. Valoracions i conclusions

En la realització del treball s'ha posat un èmfasi especial en el disseny de la interfície i dels procediments per a efectuar les mesures. El resultat final és un programa que satisfà tots els requeriments que s'havien establert en la proposta inicial i que incorpora noves funcionalitats.

Està clarament enfocat a la docència, amb menús, comandaments i procediments clars i ben estructurats. En la memòria s'ha incorporat un estudi força exhaustiu del càlcul de l'exactitud d'aquells processos de mesura en els que això era possible, amb finalitats didàctiques. En aquest sentit, doncs, el programa es posa a disposició de la Universitat de Vic per a la seva lliure utilització.

La interfície del programa és clara i neta, i destina molt espai a la imatge que s'analitza. L'estructura i disposició dels menús i dels comandaments ajuda a que la utilització del programa sigui fàcil i intuïtiva. Atès que es tracta d'un programa d'anàlisi d'imatges, i que totes les mesures s'efectuen directament sobre la imatge, aquesta estructura facilita molt la feina i contribueix a millorar l'exactitud de les mesures.

El programa s'ha estructurat de manera que sigui fàcilment ampliable amb altres rutines de mesura, o amb l'automatització de les rutines existents. En una segona fase el programa es documentarà de manera exhaustiva i es depurarà amb la finalitat de minimitzar les variables utilitzades i millorar la velocitat de funcionament.

Al tractar-se d'un programa que funciona com un instrument de mesura, es dedica un capítol sencer a desenvolupar procediments de càlcul dels errors que s'ocasionen durant la seva utilització, amb la finalitat de conèixer el seu ordre de magnitud, i de saber-los calcular de nou en cas que variïn les condicions d'utilització.

Pel que fa referència a la programació, malgrat que MATLAB no sigui un entorn de programació clàssic, sí que incorpora eines que permeten fer aplicacions no massa complexes, i orientades bàsicament als gràfics o a les imatges. L'eina GUIDE simplifica la realització de la interfície d'usuari, malgrat que presenta problemes per tractar dissenys una mica complexos. Per altra banda, el codi generat per GUIDE no és accessible, cosa que no permet modificar manualment la interfície en aquells casos en els que GUIDE té problemes. Malgrat aquests petits problemes, la potència de MATLAB compensa sobradament aquestes deficiències.

9. Bibliografia

[1] THE MATHWORKS, INC. *MATLAB: The Language of Technical Computing*. 2006. http://www.mathworks.com/help/pdf_doc/matlab/getstart.pdf (Consulta: 1 de juliol de 2010).

[2] THE MATHWORKS, INC. *MATLAB 7: Getting Started Guide*. 2010. http://www.mathworks.com/help/pdf_doc/matlab/getstart.pdf (Consulta: 1 de juliol de 2010).

[3] THE MATHWORKS, INC. *MATLAB R2010b:Documentation*. 2010. http://www.mathworks.com/help/techdoc/ (Consulta: 1 de juliol de 2010).

[4] THE MATHWORKS, INC. *MATLAB Image Processing Toolbox 7: User's Guide*. 2010. http://www.mathworks.com/help/pdf_doc/images/images_tb.pdf (Consulta: 1 de juliol de 2010).

[5] THE MATHWORKS, INC. *MATLAB Compiler 4: User's Guide*. 2010. http://www.mathworks.com/help/pdf_doc/compiler/compiler.pdf (Consulta: 24 d'agost de 2010).

[6] THE MATHWORKS, INC. *MATLAB Builder EX1: User Guide*. 2010. http://www.mathworks.com/help/pdf_doc/matlabxl/matlabxl.pdf (Consulta: 24 d'agost de 2010).

[7] GARCÍA DE JALÓN, Javier; RODRÍGUEZ, José Ignacio; VIDAL, Jesús. *Aprenda MATLAB 7.0 como si estuviera en primero*. Madrid: E. T. S. de Ingenieros Industriales, Universidad Politécnica de Madrid, 2005.

[8] BARRAGÁN GUERRERO, Diego Orlando. *Manual de interfaz gráfica de usuario en MATLAB*. 2008. http://www.matpic.com, (Consulta: 6 de juliol de 2010).

[9] ESQUEDA ELIZONDO, José Jaime. *Interfaces Gráficas en MATLAB usando GUIDE*. Méjico: Instituto Tecnológico de Ciudad Madero, Universidad Autónoma de Baja California, Tijuana, 2002.

[10] ALTMAN, Yair. *Undocumented MATLAB*. 2010. http://undocumentedmatlab.com/ (Consulta: entre l'1 de juliol i l'1 de setembre de 2010).

.

[11] GOOGLE. *Comp.soft-sys.matlab*. 2010.
http://groups.google.es/group/comp.soft-sys.matlab/topics
(Consulta: entre l'1 de juliol i l'1 de setembre de 2010).

[12] THE MATHWORKS, INC. *MATLAB Central*. 2010.
http://www.mathworks.com/matlabcentral/fileexchange
(Consulta: entre l'1 de juliol i l'1 de setembre de 2010).

[13] THE MATHWORKS, INC. *MATLAB Central Blogs*. 2010.
http://blogs.mathworks.com/
(Consulta: entre l'1 de juliol i l'1 de setembre de 2010).

[14] AENOR. *UNE-EN ISO 643: Determinación micrográfica del tamaño de grano aparente.* Madrid: AENOR, 2004.

[15] *Pautes per a la confecció de la memòria del Treball Final de Màster*.
Vic: Universitat de Vic, 2010.

10. Llistat del programa PRANIMET

```
function varargout = PRANIMET(varargin)
% PRANIMET M-file for PRANIMET.fig
%      PRANIMET, by itself, creates a new PRANIMET or raises the existing
%      singleton*.
%
%      H = PRANIMET returns the handle to a new PRANIMET or the handle to
%      the existing singleton*.
%
%      PRANIMET('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PRANIMET.M with the given input arguments.
%
%      PRANIMET('Property','Value',...) creates a new PRANIMET or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before PRANIMET_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to PRANIMET_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 PRANIMET
% Last Modified by GUIDE v2.5 18-Sep-2010 20:06:15
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @PRANIMET_OpeningFcn, ...
                   'gui_OutputFcn',  @PRANIMET_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --------------------------------------------------------------------
function varargout = PRANIMET_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

% --------------------------------------------------------------------
function figure1_CreateFcn(hObject, eventdata, handles)
global op;
try
```

```
    load('opcions.mat');
catch me
    op(1).c='b';
    op(2).c='w';op(2).w=2;op(2).s='--';op(2).m='x';op(2).ms=12;
    op(3).c='w';op(3).w=1;op(3).s='-';op(3).m='+';op(3).ms=12;
    op(4).c='w';op(4).w=12;op(4).s='';op(4).m='';op(4).ms='';
    op(5).c='r';op(5).w=2;op(5).s='-';op(5).m='+';op(5).ms=12;
    op(6).c='y';op(6).w=2;op(6).s='--';op(6).m='+';op(6).ms=12;
    op(7).c='y';op(7).w=12;op(7).s='';op(7).m='';op(7).ms='';
    op(8).c='b';
    op(9).c='b';op(9).w=2;op(9).s='-';op(9).m='+';op(9).ms=12;
    op(10).c='b';op(10).w=14;op(10).s='';op(10).m='';op(10).ms='';
    op(11).c='r';op(11).w=14;op(11).s='';op(11).m='';op(11).ms='';
end

% --------------------------------------------------------------------
function PRANIMET_OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for PRANIMET
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes PRANIMET wait for user response (see UIRESUME)
% Elements visibles a l'inici
global escala;
global calibrat;
global z;
global zoo;
global taula_part;
global S_duresa;
axes(handles.portada);
portada=imshow(imread('portada.png'));
set(handles.portada,'Visible','on');
axis off;
axes(handles.axes1);
escala=1;
S_duresa=1;
calibrat=false;
z=false;
zoo=false;
taula_part(1,1)=0;
taula_part(1,2)=0;
taula_part(1,3)=0;
set(handles.axes1,'Visible','off');
axis off;

% --------------------------------------------------------------------
function panels_invisibles(handles)
set(handles.info_foto_panel,'Visible','off');
set(handles.calibrar_panel,'Visible','off');
set(handles.mida_gra_panel,'Visible','off');
set(handles.mida_gra_iso_panel,'Visible','off');
set(handles.duresa_panel,'Visible','off');
set(handles.quantificacio_panel,'Visible','off');
```

```
set(handles.configuracio_panel,'Visible','off');
set(handles.compensacio_illuminacio_panel,'Visible','off');

% --------------------------------------------------------------------
function axes1_CreateFcn(hObject, eventdata, handles)
axis off;


% ************************************************************************
% ***** MENÚ IMATGE ******************************************************
% ************************************************************************
% --------------------------------------------------------------------
function menu_fitxer_obrir_Callback(hObject, eventdata, handles)
global PathName;      % Pathname
global FileName;      % Filename
global imatge;        % imatge
global f;
global c;
global z;
global calibrat;
global h;
[FileName,PathName] = uigetfile('*.*','Entra el nom del fitxer');
if (FileName ~= 0)
   FileName1 = sprintf('%s%s',PathName,FileName);
end
axes(handles.axes1);
try
   imatge = imread(FileName1);
   h=imshow(imatge);
   axis image;
   axis off;
   [f,c,z]=size(imatge);
   if calibrat
      dibuixar_escala();
   end
   set(handles.info_foto_text_nom_foto,'String',FileName);
   panels_invisibles(handles);
   set(handles.info_foto_panel,'Visible','on');
   try
      delete(handles.portada);
   catch me
   end
catch me
   msgbox('No es pot llegir la imatge','Error','error');
end

% --------------------------------------------------------------------
function menu_fitxer_tancar_Callback(hObject, eventdata, handles)
global h;
tancar=questdlg('Estàs segur que vols tancar la imatge?','Tancar la imatge','Si','No','default');
switch tancar
   case 'Si'
      panels_invisibles(handles);
```

```
      try
       delete(h);
      catch me
      end
   case 'No'
   %no facis res
end

% --------------------------------------------------------------------
function menu_fitxer_imprimir_automatic_Callback(hObject, eventdata, handles)
newfigure = figure;
newaxes = copyobj(handles.axes1,newfigure);
set(gcf,'PaperPositionMode','auto','PaperType','A4')
set(gcf,'Units','centimeters');
set(gcf,'PaperOrientation','landscape','InvertHardcopy','off');
set(gcf,'PaperPosition',[2,2,25,17]);
print(newfigure);
close(newfigure);

% --------------------------------------------------------------------
function menu_fitxer_imprimir_manual_Callback(hObject, eventdata, handles)
printpreview

% --------------------------------------------------------------------
function menu_fitxer_desar_Callback(hObject, eventdata, handles)
global imatge;
[filename, pathname] = uiputfile('imatge1.jpg','Desa la imatge');
if (filename ~= 0)
   filename1 = sprintf('%s%s',pathname,filename);
end
imwrite(imatge,filename1);

% --------------------------------------------------------------------
function menu_fitxer_sortir_Callback(hObject, eventdata, handles)
sortida = questdlg('Estàs segur que vols sortir?','Sortir del programa','Si','No','default');
switch sortida
   case 'Si'
   openDAQ=daqfind;
   for i=1:length(openDAQ),
      waittilstop(openDAQ(i),5);
      stop(openDAQ(i));
   end
   set(0,'ShowHiddenHandles','off')
   delete(get(0,'Children'));
   case 'No'
end


% ************************************************************************
% ***** MENU DE CALIBRAT ************************************************
% ************************************************************************
% --------------------------------------------------------------------
function menu_calibrar_calibrar_Callback(hObject, eventdata, handles)
```

```
global hc;
global hle;
global hte;
global hle2;
global op;
panels_invisibles(handles);
set(handles.calibrar_panel,'Visible','on');
set(handles.info_foto_panel,'Visible','on');
try
   delete (hle);
   delete (hte);
   delete (hle2);
catch me
end
hc=imline;
setColor(hc,op(1).c);
position = wait(hc);

% --------------------------------------------------------------------
function calibrar_edit_micres_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
set(hObject,'String','');

% --------------------------------------------------------------------
function calibrar_edit_micres_Callback(hObject, eventdata, handles)
global micres;
micres = get(handles.calibrar_edit_micres,'String');
try
   micres = str2double(micres);
catch ME
end
if (isnan(micres) | micres>50000 | micres<10)
   msgbox('Valor del micròmetre incorrecte','Warning','warn');
end

% --------------------------------------------------------------------
function calibrar_pushbutton_calibra_Callback(hObject, eventdata, handles)
global f;
global c;
global hc;
global hle2;
global micres;
global escala;
global valor_escala;
global micres_escala;
global calibrat;
global op;
liniacal=getPosition(hc);
delete(hc);
liniacal(2,2)=liniacal(1,2);
hle2=line([liniacal(1,1) liniacal(2,1)],[liniacal(1,2) liniacal(1,2)],...
```

```
    'Color',op(2).c,'LineWidth',op(2).w,'LineStyle',op(2).s,'Marker',op(2).m,...
    'MarkerSize',op(2).ms,'Tag','linia_calibrar');
escala=micres/(liniacal(2,1)-liniacal(1,1));
if escala>=10
    valor_escala(2)=100+10000/escala;
    micres_escala='10000';
elseif escala>=1
    valor_escala(2)=100+1000/escala;
    micres_escala='1000';
elseif escala>=0.1
    valor_escala(2)=100+100/escala;
    micres_escala='100';
else
    valor_escala(2)=100+10/escala;
    micres_escala='10';
end
valor_escala(1)=100;
valor_escala(3)=f-60;
valor_escala(4)=round(valor_escala(2)/2);
try
    dibuixar_escala();
catch me
end
calibrat=true;
Dim_X=num2str(round(c*escala));
Dim_Y=num2str(round(f*escala));
set(handles.calibrar_text_escala,'String',num2str(escala));
set(handles.calibrar_text_dim_x,'String',Dim_X);
set(handles.calibrar_text_dim_y,'String',Dim_Y);
set(handles.info_foto_text_valor_calibracio,'String',num2str(escala));
set(handles.info_foto_text_valor_X,'String',Dim_X);
set(handles.info_foto_text_valor_Y,'String',Dim_Y);
return

% --------------------------------------------------------------------
function dibuixar_escala()
global valor_escala;
global micres_escala;
global op;
global hle;
global hte;
hle=line([valor_escala(1) valor_escala(2)],[valor_escala(3) valor_escala(3)],...
    'Color',op(3).c,'LineWidth',op(3).w,'LineStyle',op(3).s,'Marker',op(3).m,...
    'MarkerSize',op(3).ms,'Tag','linia_escala');
hte=text(valor_escala(4),valor_escala(3)+30,sprintf('%s µm',micres_escala),...
    'Color',op(4).c,'FontSize',op(4).w,'Tag','text_escala');


% ************************************************************************
% ***** MENÚ DE MESURAR LA DURESA ****************************************
% ************************************************************************
% --------------------------------------------------------------------
function menu_mesures_duresa_Callback(hObject, eventdata, handles)
```

```
global f;
global c;
global hd;
global op;
panels_invisibles(handles);
set(handles.info_foto_panel,'Visible','on');
set(handles.duresa_panel,'Visible','on');
axes(handles.axes1);
try
   dibuixar_escala();
catch me
end
hd = imellipse(gca, [round(c/2)-200 round(f/2)-200 400 400]);
setColor(hd,op(8).c);
fcn = makeConstrainToRectFcn('imellipse',get(gca,'XLim'),get(gca,'YLim'));
setPositionConstraintFcn(hd,fcn);
setFixedAspectRatioMode(hd,1);

% --------------------------------------------------------------------
function duresa_edit_diametre_punta_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
set(hObject,'String','10');

% --------------------------------------------------------------------
function duresa_edit_pressio_punta_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
set(hObject,'String','500');

% --------------------------------------------------------------------
function duresa_pushbutton_calcular_Callback(hObject, eventdata, handles)
global hd;
global escala;
global S_duresa;
diametre_punta = get(handles.duresa_edit_diametre_punta,'String');
try
   diametre_punta = str2double(diametre_punta);
catch ME
end
if (isnan(diametre_punta) | diametre_punta>25 | diametre_punta<1)
   msgbox('Valor del diàmetre de la punta del duròmetre incorrecte','Warning','warn');
end
pressio_punta = get(handles.duresa_edit_pressio_punta,'String');
try
   pressio_punta = str2double(pressio_punta);
catch ME
end
if (isnan(pressio_punta) | pressio_punta>1000 | pressio_punta<50)
   msgbox('Valor de la pressió de la punta del duròmetre incorrecte','Warning','warn');
end
```

```
pos=getPosition(hd);
diametre_empremta=pos(1,4)/1000*escala;
duresa_brinell=2*pressio_punta/(pi*diametre_punta*(diametre_punta-(diametre_punta^2-
diametre_empremta^2)^0.5));
duresa_vickers=1.854*pressio_punta/(diametre_empremta^2);
resistencia_acer=3.5*duresa_brinell;
set(handles.duresa_text_diametre_empremta,'String',num2str(diametre_empremta));
if S_duresa==1
   set(handles.duresa_text_brinell,'String',sprintf('%0.2f',duresa_brinell));
   set(handles.duresa_text_vickers,'String',sprintf('%0.2f',duresa_vickers));
   set(handles.duresa_text_resistencia_acer,'String',sprintf('%0.2f',resistencia_acer));
end


% ************************************************************************
% ***** MENÚ DE MESURA MANUAL DEL GRA ********************************
% ************************************************************************
% ------------------------------------------------------------------
function menu_mesures_mida_gra_manual_Callback(hObject, eventdata, handles)
global imatge;
global undo_mida_gra;
global imatge_original;
global escala;
global t1;
global hFH;
global num_gra;
global num_imatge;
global taula_grans;
global bucle;
global f;
global c;
global IMATGES;
axes(handles.axes1);
try
   dibuixar_escala();
catch me
end
taula_grans=zeros(1,3);
imatge_original=imatge;
undo_mida_gra=imatge;
panels_invisibles(handles);
set(t1,'Data',taula_grans);
set(handles.mida_gra_text_diametre_gra,'String',num2str(0));
set(handles.mida_gra_text_index_m,'String',num2str(0));
set(handles.mida_gra_text_index_G,'String',num2str(0));
set(handles.mida_gra_panel,'Visible','on');
set(handles.info_foto_panel,'Visible','on');
sup_unitaria_pixel=escala*escala;
bucle=1;
num_gra=1;
num_imatge=1;
IMATGES=uint8(zeros(f,c,20));
while bucle==1
```

```
mascara=IMATGES(:,:,num_imatge);
imatgep(:,:,1)=imatge(:,:,1)+uint8(mascara);
imatgep(:,:,2)=imatge(:,:,2)+uint8(mascara);
imatgep(:,:,3)=imatge(:,:,3)+uint8(mascara);
imshow(imatgep);
axis image;
axis off;
try
   hFH = imfreehand;   % Permet resseguir un gra a mà alçada
catch me               % amb el ratolí
   break
end
if bucle~=1
   break
end
setColor(hFH,[0 1 0]);
binaryImage = hFH.createMask();   % Crea la màscara binària
% Suma tots els '1' i calcula la superfície total del gra
sup_gra=sum(sum(binaryImage==1))*sup_unitaria_pixel;
taula_grans(num_gra,1)=num_gra;   % Crea una nova fila a la taula
taula_grans(num_gra,2)=sup_gra;
% Calcula el diàmetre aparent del gra
taula_grans(num_gra,3)=round(2*(sup_gra/pi)^0.5);
set(t1,'Data',taula_grans);
sup_total=sum(taula_grans,1);
sup_aparent=sup_total(1,2)/num_gra;
diam_aparent=round(2*(sup_aparent/pi)^0.5);
num_grans_mm2=1000000/sup_aparent;
index_g=(log10(num_grans_mm2)/0.301030)-3;
set(handles.mida_gra_text_diametre_gra,'String',num2str(diam_aparent));
set(handles.mida_gra_text_index_m,'String',num2str(num_grans_mm2));
set(handles.mida_gra_text_index_G,'String',num2str(index_g));
b=uint8(binaryImage);
l=graythresh(b);
b=im2bw(b,l);
m=regionprops(b, 'centroid');
centroids = uint16(cat(1, m.Centroid));
if size(centroids)==[0,0]
   centroids(1,1)=21;
   centroids(1,2)=22;
end
if centroids(1,1)<21
   centroids(1,1)=21;
end
if centroids(1,2)<22
   centroids(1,2)=22;
end
se1 = strel('square',3);
erodedBW = imerode(b,se1);
erodedBW = imerode(erodedBW,se1);
erodedBW = imerode(erodedBW,se1);
erodedBW = imerode(erodedBW,se1);
erodedBW = imerode(erodedBW,se1);
```

```
    erodedBW = imerode(erodedBW,se1);
    erodedBW = imerode(erodedBW,se1);
    erodedBW = imerode(erodedBW,se1);
    contorn=b-erodedBW;
    s=num2str(num_gra);
    mt=text2im(s);
    [fmt cmt]=size(mt);
    mt=imresize(mt,[fmt*2 cmt*1.5]);
    [fmt cmt]=size(mt);
    contorn(centroids(1,2)-20:centroids(1,2)-21+fmt,centroids(1,1)-20:centroids(1,1)-21+cmt)=mt;
    contorn=contorn*254;
    IMATGES(:,:,num_gra+1)=IMATGES(:,:,num_gra)+uint8(contorn);
    num_gra=num_gra+1;
    num_imatge=num_imatge+1;
end

% --------------------------------------------------------------------
function mida_gra_pushbutton_acabar_Callback(hObject, eventdata, handles)
global bucle;
global hFH;
bucle=0;

% --------------------------------------------------------------------
function mida_gra_pushbutton_esborrar_Callback(hObject, eventdata, handles)
global num_gra;
global taula_grans;
global t1;
global num_imatge;
global imatge;
global IMATGES;
if num_gra==1
    return;
else
    num_gra=num_gra-1;
    num_imatge=num_imatge-1;
    taula_grans(num_gra,1)=0;
    taula_grans(num_gra,2)=0;
    taula_grans(num_gra,3)=0;
    set(t1,'Data',taula_grans);
    sup_total=sum(taula_grans,1);
    sup_aparent=sup_total(1,2)/(num_gra-1);
    diam_aparent=round(2*(sup_aparent/pi)^0.5);
    num_grans_mm2=1000000/sup_aparent;
    index_g=(log10(num_grans_mm2)/0.301030)-3;
    set(handles.mida_gra_text_diametre_gra,'String',num2str(diam_aparent));
    set(handles.mida_gra_text_index_m,'String',num2str(num_grans_mm2));
    set(handles.mida_gra_text_index_G,'String',num2str(index_g));
    mascara=IMATGES(:,:,num_imatge);
    imatgep(:,:,1)=imatge(:,:,1)+uint8(mascara);
    imatgep(:,:,2)=imatge(:,:,2)+uint8(mascara);
    imatgep(:,:,3)=imatge(:,:,3)+uint8(mascara);
    try
        dibuixar_escala();
```

```
    catch me
    end
    axis equal;
    axis off;
end

% --------------------------------------------------------------------
function mida_gra_taula_1_CreateFcn(hObject, eventdata, handles)
global t1
t1=(hObject);

% --------------------------------------------------------------------
function mida_gra_taula_1_CellEditCallback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function imtext=text2im(text)
% text2im - converteix un text numèric en imatge
text=text+0;                % converting string into Ascii-number array
laenge=length(text);
imtext=zeros(20,18*laenge);    % Preparing the resulting image
for i=1:laenge
    code=text(i);           % Ascii code of i.th letter
    if code==48
    TxtIm=[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
     1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
     1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1;
     1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1;
     1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
     1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
     1,1,0,0,0,0,1,1,1,1,1,0,0,0,0,0,1,1;
     1,1,0,0,0,0,1,1,1,1,0,0,0,0,0,0,1,1;
     1,1,0,0,0,0,1,1,0,0,1,1,0,0,0,0,1,1;
     1,1,0,0,0,0,1,1,0,0,1,1,0,0,0,0,1,1;
     1,1,0,0,0,0,0,0,1,1,1,1,0,0,0,0,1,1;
     1,1,0,0,0,0,0,0,1,1,1,1,0,0,0,0,1,1;
     1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
     1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
     1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
     1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1;
     1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1;
     1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
     1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1];
    elseif code==49
    TxtIm=[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
     1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
     1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1;
     1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1;
     1,1,1,1,1,0,0,0,0,0,0,1,1,1,1,1,1,1;
     1,1,1,1,1,0,0,0,0,0,0,1,1,1,1,1,1,1;
     1,1,1,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1;
     1,1,1,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1;
     1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1;
```

```
 1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1;
 1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1;
 1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1;
 1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1;
 1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1;
 1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1;
 1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1;
 1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1;
 1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1];
elseif code==50
TxtIm=[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
 1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1;
 1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1;
 1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1;
 1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1];
elseif code==51
TxtIm=[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
 1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1;
 1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1];
```

```
elseif code==52
TxtIm=[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1;
 1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,1,1,1,1;
 1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,1,1,1,1;
 1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,1,1,1,1;
 1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,1,1,1,1;
 1,1,1,1,0,0,0,0,1,1,0,0,0,0,1,1,1,1;
 1,1,1,1,0,0,0,0,1,1,0,0,0,0,1,1,1,1;
 1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,1;
 1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,1;
 1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1;
 1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1;
 1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1];
elseif code==53
TxtIm=[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
 1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1;
 1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1;
 1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1;
 1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1;
 1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1];
elseif code==54
TxtIm=[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
 1,1,1,1,1,1,0,0,0,0,0,0,0,1,1,1,1;
 1,1,1,1,1,1,0,0,0,0,0,0,0,1,1,1,1;
 1,1,1,1,0,0,0,0,1,1,1,1,1,1,1,1,1;
 1,1,1,1,0,0,0,0,1,1,1,1,1,1,1,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1;
 1,1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1;
 1,1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1;
```

```
 1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1;
 1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1];
elseif code==55
TxtIm=[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
 1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1;
 1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1;
 1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1;
 1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1;
 1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1;
 1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1;
 1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1;
 1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1;
 1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1;
 1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1;
 1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1];
elseif code==56
TxtIm=[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
 1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1;
 1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1;
 1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
 1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1;
 1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1];
elseif code==57
```

```
    TxtIm=[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
     1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
     1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1;
     1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1;
     1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
     1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
     1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
     1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1;
     1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1;
     1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1;
     1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1;
     1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1;
     1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1;
     1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1;
     1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1;
     1,1,1,1,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1;
     1,1,1,1,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1;
     1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
     1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1];
   end
   imtext(1:20,((i-1)*18+1):i*18)=TxtIm;
end
```

```
% ************************************************************************
% ***** MENÚ DE MESURA EL GRA UNE-ISO ************************************
% ************************************************************************
% ------------------------------------------------------------------------
function menu_mesures_mida_gra_iso_Callback(hObject, eventdata, handles)
global c;
global f;
global num_gra_interior;
global num_gra_frontera;
global bucle_i;
global bucle_f;
global surt;
global hti;
global htf;
global hq;
panels_invisibles(handles);
set(handles.mida_gra_iso_text_diametre_gra,'String',sprintf('%0.2f',0));
set(handles.mida_gra_iso_text_index_m,'String',sprintf('%0.2f',0));
set(handles.mida_gra_iso_text_index_G,'String',sprintf('%0.2f',0));
set(handles.iso_mesura_grans_text_interiors,'String',sprintf('%d',0));
set(handles.iso_mesura_grans_text_frontera,'String',sprintf('%d',0));
set(handles.mida_gra_iso_panel,'Visible','on');
set(handles.info_foto_panel,'Visible','on');
try
   delete(hti);
```

```
   delete(htf);
catch me
end
axes(handles.axes1);
try
   dibuixar_escala();
catch me
end
hq = imrect(gca, [round(c/2)-500 round(f/2)-500 1000 1000]);
addNewPositionCallback(hq,@(p) title(mat2str(p,3)));
fcn = makeConstrainToRectFcn('imrect',get(gca,'XLim'),get(gca,'YLim'));
setPositionConstraintFcn(hq,fcn);
setFixedAspectRatioMode(hq,1);
bucle_i=1;
bucle_f=1;
num_gra_interior=1;
num_gra_frontera=1;
surt=1;

% --------------------------------------------------------------------
function iso_mesura_grans_pushbutton_fixar_marc_Callback(hObject, eventdata, handles)
global hq;
global pos;
global op;
global surt;
pos=getPosition(hq);
delete(hq);
line([pos(1) pos(1)+pos(3)],[pos(2) pos(2)],'Color',op(9).c,'LineWidth',op(9).w,...
   'LineStyle',op(9).s,'Marker',op(9).m,'MarkerSize',op(9).ms,'Tag','linia_iso');
line([pos(1) pos(1)+pos(3)],[pos(2)+pos(4) pos(2)+pos(4)],'Color',op(9).c,'LineWidth',op(9).w,...
   'LineStyle',op(9).s,'Marker',op(9).m,'MarkerSize',op(9).ms,'Tag','linia_iso');
line([pos(1)+pos(3) pos(1)+pos(3)],[pos(2) pos(2)+pos(4)],'Color',op(9).c,'LineWidth',op(9).w,...
   'LineStyle',op(9).s,'Marker',op(9).m,'MarkerSize',op(9).ms,'Tag','linia_iso');
line([pos(1) pos(1)],[pos(2) pos(2)+pos(4)],'Color',op(9).c,'LineWidth',op(9).w,...
   'LineStyle',op(9).s,'Marker',op(9).m,'MarkerSize',op(9).ms,'Tag','linia_iso');
surt=0;

% --------------------------------------------------------------------
function iso_mesura_grans_pushbutton_interior_Callback(hObject, eventdata, handles)
global bucle_i;
global bucle_f;
global num_gra_interior;
global surt;
global hti;
global f;
global c;
global op;
bucle_i=1;
bucle_f=0;
while bucle_i==1   % Fes, mentre no premis el botó Grans Frontera
   bucle_f=0;     % Inactiva la funció Grans Frontera
   try
      k = waitforbuttonpress;    % Espera que l'usuari premi el ratolí
```

```
  catch me
  end
  if surt==1
    break
  end
  point1 = get(gca,'CurrentPoint');    % Detecta la pulsació del botó del ratolí
  point1 = point1(1,1:2);              % Extreu les coordenades del punt
  if point1(1)>=0 & point1(1)<=c & point1(2)>=0 & point1(2)<=f
    if num_gra_interior>9        % Comprova que el ratolí no estigui fora de
      point1(1)=point1(1)-20;    % la imatge.
    else
      point1(1)=point1(1)-10;
    end
    % Crea un nou objecte gràfic, el text corresponent al número del
    % nou gra, i el mostra per la pantalla en la posició del ratolí.
    % Al mateix temps, desa el handle d'aquest objecte en una matriu,
    % per si cal esborrar-lo
    hti(num_gra_interior)=text(point1(1),point1(2),sprintf('%0.f',num_gra_interior),...
      'Color',op(10).c,'FontSize',op(10).w,'Tag','text_mesura_interior');
    set(handles.iso_mesura_grans_text_interiors,'String',sprintf('%d',num_gra_interior));
    num_gra_interior=num_gra_interior+1;   % Incrementa el número del gra
  end
end

% --------------------------------------------------------------------
function iso_mesura_grans_pushbutton_frontera_Callback(hObject, eventdata, handles)
global bucle_i;
global bucle_f;
global num_gra_frontera;
global surt;
global htf;
global f;
global c;
global op;
bucle_i=0;
bucle_f=1;
while bucle_f==1
  bucle_i=0;
  try
    k = waitforbuttonpress;
  catch me
  end
  if surt==1
    break
  end
  point1 = get(gca,'CurrentPoint');    % detecta la pulsació del botó del ratolí
  point1 = point1(1,1:2);              % extreu les coordenades del punt
  if point1(1)>=0 & point1(1)<=c & point1(2)>=0 & point1(2)<=f
    if num_gra_frontera>9
      point1(1)=point1(1)-20;
    else
      point1(1)=point1(1)-10;
    end
```

```
      htf(num_gra_frontera)=text(point1(1),point1(2),sprintf('%0.f',num_gra_frontera),...
        'Color',op(11).c,'FontSize',op(11).w,'Tag','text_mesura_frontera');
      set(handles.iso_mesura_grans_text_frontera,'String',sprintf('%d',num_gra_frontera));
      num_gra_frontera=num_gra_frontera+1;
    end
end

% --------------------------------------------------------------------
function iso_mesura_grans_pushbutton_esborrar_Callback(hObject, eventdata, handles)
global num_gra_interior;
global num_gra_frontera;
global bucle_i;
global hti;
global htf;
if bucle_i==1
   delete(hti(num_gra_interior-1));
   num_gra_interior=num_gra_interior-1;
   set(handles.iso_mesura_grans_text_interiors,'String',sprintf('%d',num_gra_interior-1));
else
   delete(htf(num_gra_frontera-1));
   num_gra_frontera=num_gra_frontera-1;
   set(handles.iso_mesura_grans_text_frontera,'String',sprintf('%d',num_gra_frontera-1));
end
return

% --------------------------------------------------------------------
function iso_mesura_grans_pushbutton_calcular_Callback(hObject, eventdata, handles)
global num_gra_interior;
global num_gra_frontera;
global bucle_i;
global bucle_f;
global escala;
global pos;
global surt;
set(handles.iso_mesura_grans_text_frontera,'String',sprintf('%d',num_gra_frontera-1));
set(handles.iso_mesura_grans_text_interiors,'String',sprintf('%d',num_gra_interior-1));
bucle_i=0;
bucle_f=0;
sup_quadrat=pos(3)*pos(4)*escala^2;
num_grans=(num_gra_interior-1)+(num_gra_frontera-1)/2+1;
num_grans_mm2=num_grans/(sup_quadrat/1000000);
diam_aparent=(((sup_quadrat/num_grans_mm2)/pi)^0.5)*2;
index_g=(log10(num_grans_mm2)/0.301030)-3;
set(handles.mida_gra_iso_text_diametre_gra,'String',sprintf('%0.2f',diam_aparent));
set(handles.mida_gra_iso_text_index_m,'String',sprintf('%0.2f',num_grans_mm2));
set(handles.mida_gra_iso_text_index_G,'String',sprintf('%0.2f',index_g));
surt=1;


% ************************************************************************
% ***** MENÚ DE QUANTIFICAR LES FASES ************************************
% ************************************************************************
% --------------------------------------------------------------------
```

```
function menu_quantificacio_Callback(hObject, eventdata, handles)
global h2;
global imatge;
global imatge_o;
global imatge_q;
global div_pant;
div_pant=0;
panels_invisibles(handles);
set(handles.quantificacio_panel,'Visible','on');
set(handles.info_foto_panel,'Visible','on');
set(handles.quantificacio_slider_binaritzacio,'value',0.5);
axes(handles.axes2);
set(handles.axes2,'Visible','on');
h2=imshow(imatge);
axis image;
axis off;
axes(handles.axes1);
set(handles.axes1,'Visible','on');
imatge_q=imatge;
imatge_o=imatge;
imshow(imatge_q);
try
   dibuixar_escala();
catch me
end
axis image;
axis off;

% --------------------------------------------------------------------
function quantificacio_slider_brillantor_Callback(hObject, eventdata, handles)
global imatge_q;
global imatge_o;
a=get(hObject,'Value');
imatge_q=double(imatge_o)+(a*255);
imatge_q=uint8(imatge_q);
imshow(imatge_q);
axis image;
axis off;

% --------------------------------------------------------------------
function quantificacio_slider_brillantor_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --------------------------------------------------------------------
function quantificacio_pushbutton_grisos_Callback(hObject, eventdata, handles)
global imatge_q;
global imatge_o;
global z;
[x y z]=size(imatge_q);
if z==3
   imatge_q=rgb2gray(imatge_q);
```

```
end
imshow(imatge_q);
axis image;
axis off;
imatge_o=imatge_q;

% --------------------------------------------------------------------
function quantificacio_pushbutton_contrast_Callback(hObject, eventdata, handles)
imcontrast(gcf);

% --------------------------------------------------------------------
function quantificacio_pushbutton_imatge_original_Callback(hObject, eventdata, handles)
global imatge;
global imatge_q;
global imatge_o;
global h2;
set(handles.axes2,'Visible','on');
axes(handles.axes2);
imshow(imatge);
axis image;
axis off;
set(handles.axes1,'Visible','on');
axes(handles.axes1);
imatge_q=imatge;
imatge_o=imatge;
imshow(imatge_q);
axis image;
axis off;

% --------------------------------------------------------------------
function quantificacio_slider_binaritzacio_Callback(hObject, eventdata, handles)
global f;
global c;
global b;
global imatge_q;
global div_pant;
global imatge;
global id1;
set(handles.axes1,'Visible','on');
axes(handles.axes1);
lev=get(hObject,'Value');
b=im2bw(imatge_q,lev);
se = strel('diamond',1);
b=imopen(b,se);
if div_pant==1
   id1 =uint8(imatge);
   id2=uint8(b);
   id3=id2(:,(1:round(c/2)),1);
   id1(:,(round(c/2)+1:c),:)=id1(:,(1:round(c/2)),:);
   id1(:,(1:round(c/2)),1)=id3*255;
   id1(:,(1:round(c/2)),2)=id3*255;
   id1(:,(1:round(c/2)),3)=id3*255;
else
```

```
    id1=b;
end
imshow(id1);
axis image;
axis off;
clar=sum(sum(b));
total=f*c;
perc_clar=clar*100/double(total);
perc_fosc=100-perc_clar;
set(handles.quantificar_text_clar,'String',sprintf('%0.2f',perc_clar));
set(handles.quantificar_text_fosc,'String',sprintf('%0.2f',perc_fosc));
axes(handles.axes1);

% --------------------------------------------------------------------
function quantificacio_slider_binaritzacio_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --------------------------------------------------------------------
function quantificacio_pushbutton_dividir_pantalla_Callback(hObject, eventdata, handles)
global div_pant
global id1;
global b;
global c;
global imatge;
if div_pant==1
    div_pant=0;
    imshow(b);
else
    div_pant=1;
    id1 =uint8(imatge);
    id2=uint8(b);
    id3=id2(:,(1:round(c/2)),1);
    id1(:,(round(c/2)+1:c),:)=id1(:,(1:round(c/2)),:);
    id1(:,(1:round(c/2)),1)=id3*255;
    id1(:,(1:round(c/2)),2)=id3*255;
    id1(:,(1:round(c/2)),3)=id3*255;
    imshow(id1);
end

% --------------------------------------------------------------------
function quantificacio_pushbutton_automatic_Callback(hObject, eventdata, handles)
global imatge_q;
global b;
global d;
global f;
global c;
global x;
global y;
global q;
global S1;
global S2;
```

```
global S3;
global S4;
[x y z]=size(imatge_q);
if z==3
   imatge_q=rgb2gray(imatge_q);
end
q=imatge_q;
[x y]=size(q);
d=double(q);
if S1==1
   compensacio_automatica_illuminacio1();
end
if S2==1
   compensacio_automatica_illuminacio2();
end
if S3==1
   compensacio_automatica_illuminacio3();
end
if S4==1
   compensacio_automatica_illuminacio4();
end
imatge_q=uint8(d);
lev=graythresh(imatge_q);
b=im2bw(imatge_q,lev);
imshow(b);
axis image;
axis off;
set(handles.quantificacio_slider_binaritzacio,'Value',lev);
clar=sum(sum(b));
total=f*c;
perc_clar=clar*100/double(total);
perc_fosc=100-perc_clar;
set(handles.quantificar_text_clar,'String',sprintf('%0.2f',perc_clar));
set(handles.quantificar_text_fosc,'String',sprintf('%0.2f',perc_fosc));
axes(handles.axes1);

% --------------------------------------------------------------------
function compensacio_automatica_illuminacio1()
global f;
global c;
global d;
n=10;
o=zeros(1,n);
o=double(o);
b=d;
for i=1:n
   o(i)=sum(sum(b(1:f,round(c/n*(i-1))+1:round((c/n)*i))));
end
o1=o(1);
for i=1:n
   o(i)=ceil((o(i)-o1)*n/(f*c));
end
p=zeros(1,round(c/n));
```

```
p=[p linspace(0,o(2),c/n)];
for i=2:n-2
   p=[p linspace(o(i),o(i+1),c/n)];
end
[p1 p2]=size(p);
p=[p linspace(o(n-1),o(n),c-p2)];
prod=zeros(f,c);
for i=1:f
   prod(i,:)=p;
end
d=double(b);
d=d-double(prod);
n=8;
o=zeros(1,n);
b=d;
o=double(o);
for i=1:n
   o(i)=sum(sum(b((f/n*(i-1))+1:(f/n)*i,1:c)));
end
o1=o(1);
for i=1:n
   o(i)=ceil((o(i)-o1)*n/(f*c));
end
p=zeros(1,f/n);
p=[p linspace(0,o(2),f/n)];
for i=2:n-2
   p=[p linspace(o(i),o(i+1),f/n)];
end
[p1 p2]=size(p);
p=[p linspace(o(n-1),o(n),f-p2)];
p=p';
prod=zeros(f,c);
for i=1:c
   prod(:,i)=p;
end
d=double(b);
d=d-double(prod);

% ------------------------------------------------------------------------
function compensacio_automatica_illuminacio2()
global f;
global c;
global d;
n=10;
sum1=sum(sum(d(1:f,1:c/n)));
sum10=sum(sum(d(1:f,round((c/n*9))+1:c)));
dif=ceil((sum1-sum10)/(f*c/10));
if dif>=0
   p=linspace(dif,0,c);
else
   p=linspace(0,dif*(-1),c);
end
prod=zeros(f,c);
```

```
for i=1:f
   prod(i,:)=p;
end
d=double(d);
d1=d-double(prod);
n=10;
sum1=sum(sum(d1(1:f/n,1:c)));
sum10=sum(sum(d1(round((f/n*9))+1:f,1:c)));
dif=ceil((sum1-sum10)/(f*c/10));
if dif>=0
   p=linspace(dif,0,f)';
else
   p=linspace(0,dif*(-1),f)';
end
prod=zeros(f,c);
for i=1:c
   prod(:,i)=p;
end
d2=d1-double(prod);
d=uint8(d2);

% ------------------------------------------------------------------
function compensacio_automatica_illuminacio3()
global d;
global x;
global y;
n=16;
for i=1:n
  r(i)=sum(sum(d(1:x,round(y/n*(i-1))+1:round(y/n*i))));
end
tot=sum(r);
prom=tot/n;
for i=1:n
   d(1:x,round(y/n*(i-1))+1:round(y/n*i))=d(1:x,round(y/n*(i-1))+1:round(y/n*i))/(r(i)/prom);
end
s=0;
if s==1
   n=2;
   for i=1:n
     r(i)=sum(sum(d(round(x/n*(i-1))+1:round(x/n*i),1:y)));
   end
   tot=sum(r);
   prom=tot/n;
   for i=1:n
      d(round(x/n*(i-1))+1:round(x/n*i),1:y)=d(round(x/n*(i-1))+1:round(x/n*i),1:y)/(r(i)/prom);
   end
end
m=linspace(1.1,1,x)';
prod=zeros(x,y);
for i=1:y
   prod(:,i)=m;
end
d=double(d)./double(prod);
```

```
% --------------------------------------------------------------------
function compensacio_automatica_illuminacio4()
global d;
global x;
global y;
n=16;
r=zeros(n,n);
mi=zeros(n,n);
ma=zeros(n,n);
for j=1:n
   for i=1:n
      r(j,i)=sum(sum(d(round((x/n)*(i-1))+1:round((x/n)*i),round((y/n)*(j-1))+1:round((y/n)*j))));
      mi=min(min(d(round((x/n)*(i-1))+1:round((x/n)*i),round((y/n)*(j-1))+1:round((y/n)*j))));
      ma=max(max(d(round((x/n)*(i-1))+1:round((x/n)*i),round((y/n)*(j-1))+1:round((y/n)*j))));
      d(round((x/n)*(i-1))+1:round((x/n)*i),round((y/n)*(j-1))+1:round((y/n)*j))=...
         d(round((x/n)*(i-1))+1:round((x/n)*i),round((y/n)*(j-1))+1:round((y/n)*j))-mi;
      d(round((x/n)*(i-1))+1:round((x/n)*i),round((y/n)*(j-1))+1:round((y/n)*j))=...
         d(round((x/n)*(i-1))+1:round((x/n)*i),round((y/n)*(j-1))+1:round((y/n)*j))*255/ma;
   end
end

% --------------------------------------------------------------------
function quantificacio_pushbutton_quantificar_Callback(hObject, eventdata, handles)
global b;
global f;
global c;
clar=sum(sum(b));
total=f*c;
perc_clar=clar*100/double(total);
perc_fosc=100-perc_clar;
set(handles.quantificar_text_clar,'String',sprintf('%0.2f',perc_clar));
set(handles.quantificar_text_fosc,'String',sprintf('%0.2f',perc_fosc));
axes(handles.axes1);

% --------------------------------------------------------------------
function axes2_CreateFcn(hObject, eventdata, handles)
axis off;

% --------------------------------------------------------------------
function radiobutton1_Callback(hObject, eventdata, handles)
global S1;
global S2;
global S3;
global S4;
S1=get(hObject,'Value');
if S1==1
   S2=0;
   S3=0;
   S4=0;
   set(handles.radiobutton2,'Value',0);
   set(handles.radiobutton3,'Value',0);
   set(handles.radiobutton4,'Value',0);
```

```
end

% --------------------------------------------------------------------
function radiobutton2_Callback(hObject, eventdata, handles)
global S1;
global S2;
global S3;
global S4;
S2=get(hObject,'Value');
if S2==1
   S1=0;
   S3=0;
   S4=0;
   set(handles.radiobutton1,'Value',0);
   set(handles.radiobutton3,'Value',0);
   set(handles.radiobutton4,'Value',0);
end

% --------------------------------------------------------------------
function radiobutton3_Callback(hObject, eventdata, handles)
global S1;
global S2;
global S3;
global S4;
S3=get(hObject,'Value');
if S3==1
   S1=0;
   S2=0;
   S4=0;
   set(handles.radiobutton1,'Value',0);
   set(handles.radiobutton2,'Value',0);
   set(handles.radiobutton4,'Value',0);
end

% --------------------------------------------------------------------
function radiobutton4_Callback(hObject, eventdata, handles)
global S1;
global S2;
global S3;
global S4;
S4=get(hObject,'Value');
if S4==1
   S1=0;
   S2=0;
   S3=0;
   set(handles.radiobutton1,'Value',0);
   set(handles.radiobutton2,'Value',0);
   set(handles.radiobutton3,'Value',0);
end

% --------------------------------------------------------------------
function radiobutton1_CreateFcn(hObject, eventdata, handles)
global S1;
```

```
global S2;
global S3;
global S4;
set(hObject,'Value',1);
S1=1;
S2=0;
S3=0;
S4=0;

% --------------------------------------------------------------------
function radiobutton2_CreateFcn(hObject, eventdata, handles)
set(hObject,'Value',0);

% --------------------------------------------------------------------
function radiobutton3_CreateFcn(hObject, eventdata, handles)
set(hObject,'Value',0);

% --------------------------------------------------------------------
function radiobutton4_CreateFcn(hObject, eventdata, handles)
set(hObject,'Value',0);


% ************************************************************************
% ***** MENU COMPENSACIÓ DE LA IL·LUMINACIÓ ******************************
% ************************************************************************
% --------------------------------------------------------------------
function compensacio_illuminacio_menu_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function compensacio_illuminacio_patro_menu_Callback(hObject, eventdata, handles)
global imatge;
panels_invisibles(handles);
set(handles.compensacio_illuminacio_panel,'Visible','on');
set(handles.info_foto_panel,'Visible','on');
set(handles.compensacio_illuminacio_slider_brillantor,'value',0.5);
axes(handles.axes1);
imshow(imatge);
axis image;
axis off;

% --------------------------------------------------------------------
function compensacio_illuminacio_pushbutton_llegir_patro_Callback(hObject, eventdata, handles)
global imatge_patro;
global h4;
[FileName,PathName] = uigetfile('*.*','Entra el nom del fitxer patró');
if (FileName ~= 0)
   FileName1 = sprintf('%s%s',PathName,FileName);
end
imatge_patro = imread(FileName1);
axes(handles.axes4);
imshow(imatge_patro);
axis image;
axis off;
```

```
axes(handles.axes1);

% --------------------------------------------------------------------
function compensacio_illuminacio_pushbutton_compensar_Callback(hObject, eventdata, handles)
global imatge;
global imatge_patro;
global imatge_o;
imatge=uint8(round((double(imatge)./double(imatge_patro))*100));
axes(handles.axes1);
imshow(imatge);
axis image;
axis off;
imatge_o=imatge;

% --------------------------------------------------------------------
function compensacio_illuminacio_pushbutton_nivells_Callback(hObject, eventdata, handles)
global imatge;
global imatge_o;
m=0.5;
for k=1:3
   t=0;
   im1=imatge(:,:,k);
   x=(0:255);
   h=hist(im1(:),x);
   t=cumsum(h.^m);
   t=t.*(256/max(t));
   imatge(:,:,k)=t(imatge(:,:,k)+1);
end
imatge=uint8(round(imatge));
imshow(imatge);
imatge_o=imatge;
axis image;
axis off;

% --------------------------------------------------------------------
function compensacio_illuminacio_slider_brillantor_Callback(hObject, eventdata, handles)
global imatge;
global imatge_o;
a=get(hObject,'Value');
imatge_o=double(imatge)+((a-0.5)*255);
imshow(uint8(imatge_o));
axis image;
axis off;

% --------------------------------------------------------------------
function compensacio_illuminacio_slider_brillantor_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --------------------------------------------------------------------
function compensacio_illuminacio_pushbutton_acabar_Callback(hObject, eventdata, handles)
global imatge;
```

```
global imatge_o;
imatge=uint8(imatge_o);
set(handles.compensacio_illuminacio_panel,'Visible','off');

% --------------------------------------------------------------------
function axes4_CreateFcn(hObject, eventdata, handles)
axis off;


% ************************************************************************
% ***** MENÚ D'AJUDA *****************************************************
% ************************************************************************
% --------------------------------------------------------------------
function ajuda_menu_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function Ajuda_documentacio_menu_Callback(hObject, eventdata, handles)
%
fitxer_documentacio = 'Documentació_PAIM.pdf';
try
   open(helpfile);
catch me
   msgbox('No es troba el fitxer de documentació!','Error','Error');
end

% --------------------------------------------------------------------
function ajuda_versio_menu_Callback(hObject, eventdata, handles)




% ************************************************************************
% ***** BOTÓ DE CONFIGURACIÓ *********************************************
% ************************************************************************
% --------------------------------------------------------------------
function pop1_Callback(hObject, eventdata, handles)
%
global op;
g1=['r' 'g' 'b' 'c' 'm' 'y' 'k' 'w'];
a=get(hObject,'Value');
op(2).c=g1(a);
hh=findall(gcf,'Tag','linia_calibrar');
set(hh,'Color',op(2).c);

% --- Executes during object creation, after setting all properties.
function pop1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
global op;
g1=['r' 'g' 'b' 'c' 'm' 'y' 'k' 'w'];
set(hObject,'Value',find(g1==op(2).c));
```

```
% --- Executes on selection change in pop2.
function pop2_Callback(hObject, eventdata, handles)
global op;
g2=[1 2 3 4 10 12 14 16 18 20];
a=get(hObject,'Value');
op(2).w=g2(a);
hh=findall(gcf,'Tag','linia_calibrar');
set(hh,'LineWidth',op(2).w);

% --- Executes during object creation, after setting all properties.
function pop2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
global op;
g2=[1 2 3 4 10 12 14 16 18 20];
set(hObject,'Value',find(g2==op(2).w));

% --- Executes on selection change in pop3.
function pop3_Callback(hObject, eventdata, handles)
global op;
g3=['-' '--' ':' '-.'];
a=get(hObject,'Value');
op(2).s=g3(a);
if a==2
   op(2).s='--';
end
if a==4
   op(2).s='-.';
end
hh=findall(gcf,'Tag','linia_calibrar');
set(hh,'LineStyle',op(2).s);

% --- Executes during object creation, after setting all properties.
function pop3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
global op;
g3=['-' '=' ':' '.'];
op2=op(2).s;
if op2=='--'
   op2='=';
end
if op2=='-.'
   op2='.';
end
set(hObject,'Value',find(g3==op2));

% --- Executes on selection change in pop4.
function pop4_Callback(hObject, eventdata, handles)
global op;
```

```
g4=['+' 'o' '*' 'x' 's' 'd'];
a=get(hObject,'Value');
op(2).m=g4(a);
hh=findall(gcf,'Tag','linia_calibrar');
set(hh,'Marker',op(2).m);

% --- Executes during object creation, after setting all properties.
function pop4_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
global op;
g4=['+' 'o' '*' 'x' 's' 'd'];
set(hObject,'Value',find(g4==op(2).m));

% --- Executes on selection change in pop5.
function pop5_Callback(hObject, eventdata, handles)
global op;
g5=[10 12 14 16 18 20];
a=get(hObject,'Value');
op(2).ms=g5(a);
hh=findall(gcf,'Tag','linia_calibrar');
set(hh,'MarkerSize',op(2).ms);

% --- Executes during object creation, after setting all properties.
function pop5_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
global op;
g5=[10 12 14 16 18 20];
set(hObject,'Value',find(g5==op(2).ms));

% --- Executes on selection change in pop6.
function pop6_Callback(hObject, eventdata, handles)
global op;
g1=['r' 'g' 'b' 'c' 'm' 'y' 'k' 'w'];
a=get(hObject,'Value');
op(3).c=g1(a);
hh=findall(gcf,'Tag','linia_escala');
set(hh,'Color',op(3).c);

% --- Executes during object creation, after setting all properties.
function pop6_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
global op;
g1=['r' 'g' 'b' 'c' 'm' 'y' 'k' 'w'];
set(hObject,'Value',find(g1==op(3).c));

% --- Executes on selection change in pop7.
function pop7_Callback(hObject, eventdata, handles)
```

```
global op;
g2=[1 2 3 4 10 12 14 16 18 20];
a=get(hObject,'Value');
op(3).w=g2(a);
hh=findall(gcf,'Tag','linia_escala');
set(hh,'LineWidth',op(3).w);

% --- Executes during object creation, after setting all properties.
function pop7_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
global op;
g2=[1 2 3 4 10 12 14 16 18 20];
set(hObject,'Value',find(g2==op(3).w));

% --- Executes on selection change in pop8.
function pop8_Callback(hObject, eventdata, handles)
global op;
g3=['-' '--' ':' '-.'];
a=get(hObject,'Value');
op(3).s=g3(a);
if a==2
   op(3).s='--';
end
if a==4
   op(3).s='-.';
end
hh=findall(gcf,'Tag','linia_escala');
set(hh,'LineStyle',op(3).s);

% --- Executes during object creation, after setting all properties.
function pop8_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
global op;
g3=['-' '=' ':' '.'];
op2=op(3).s;
if op2=='--'
   op2='=';
end
if op2=='-.'
   op2='.';
end
set(hObject,'Value',find(g3==op2));

% --- Executes on selection change in pop9.
function pop9_Callback(hObject, eventdata, handles)
global op;
g4=['+' 'o' '*' 'x' 's' 'd'];
a=get(hObject,'Value');
op(3).m=g4(a);
```

```
hh=findall(gcf,'Tag','linia_escala');
set(hh,'Marker',op(3).m);

% --- Executes during object creation, after setting all properties.
function pop9_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
global op;
g4=['+' 'o' '*' 'x' 's' 'd'];
set(hObject,'Value',find(g4==op(3).m));

% --- Executes on selection change in pop10.
function pop10_Callback(hObject, eventdata, handles)
global op;
g5=[10 12 14 16 18 20];
a=get(hObject,'Value');
op(3).ms=g5(a);
hh=findall(gcf,'Tag','linia_escala');
set(hh,'MarkerSize',op(3).ms);

% --- Executes during object creation, after setting all properties.
function pop10_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
global op;
g5=[10 12 14 16 18 20];
set(hObject,'Value',find(g5==op(3).ms));

% --- Executes on selection change in pop11.
function pop11_Callback(hObject, eventdata, handles)
global op;
g1=['r' 'g' 'b' 'c' 'm' 'y' 'k' 'w'];
a=get(hObject,'Value');
op(4).c=g1(a);
hh=findall(gcf,'Tag','text_escala');
set(hh,'Color',op(4).c);

% --- Executes during object creation, after setting all properties.
function pop11_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
global op;
g1=['r' 'g' 'b' 'c' 'm' 'y' 'k' 'w'];
set(hObject,'Value',find(g1==op(4).c));

% --- Executes on selection change in pop12.
function pop12_Callback(hObject, eventdata, handles)

global op;
g2=[1 2 3 4 10 12 14 16 18 20];
```

```
a=get(hObject,'Value');
op(4).w=g2(a);
hh=findall(gcf,'Tag','text_escala');
set(hh,'FontSize',op(4).w);

% --- Executes during object creation, after setting all properties.
function pop12_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
global op;
g2=[1 2 3 4 10 12 14 16 18 20];
set(hObject,'Value',find(g2==op(4).w));

% --- Executes on selection change in pop13.
function pop13_Callback(hObject, eventdata, handles)
global op;
g1=['r' 'g' 'b' 'c' 'm' 'y' 'k' 'w'];
a=get(hObject,'Value');
op(5).c=g1(a);

% --- Executes during object creation, after setting all properties.
function pop13_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in pop14.
function pop14_Callback(hObject, eventdata, handles)
global op;
g1=['r' 'g' 'b' 'c' 'm' 'y' 'k' 'w'];
a=get(hObject,'Value');
op(6).c=g1(a);
hh=findall(gcf,'Tag','linia_mesura');
set(hh,'Color',op(6).c);

% --- Executes during object creation, after setting all properties.
function pop14_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
global op;
g1=['r' 'g' 'b' 'c' 'm' 'y' 'k' 'w'];
set(hObject,'Value',find(g1==op(6).c));

% --- Executes on selection change in pop15.
function pop15_Callback(hObject, eventdata, handles)
global op;
g2=[1 2 3 4 10 12 14 16 18 20];
a=get(hObject,'Value');
op(6).w=g2(a);
hh=findall(gcf,'Tag','linia_mesura');
set(hh,'LineWidth',op(6).w);
```

```
% --- Executes during object creation, after setting all properties.
function pop15_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
global op;
g2=[1 2 3 4 10 12 14 16 18 20];
set(hObject,'Value',find(g2==op(6).w));

% --- Executes on selection change in pop16.
function pop16_Callback(hObject, eventdata, handles)
global op;
g3=['-' '--' ':' '-.'];
a=get(hObject,'Value');
op(6).s=g3(a);
if a==2
   op(6).s='--';
end
if a==4
   op(6).s='-.';
end
hh=findall(gcf,'Tag','linia_mesura');
set(hh,'LineStyle',op(6).s);

% --- Executes during object creation, after setting all properties.
function pop16_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
global op;
g3=['-' '=' ':' '.'];
op2=op(6).s;
if op2=='--'
   op2='=';
end
if op2=='-.'
   op2='.';
end
set(hObject,'Value',find(g3==op2));

% --- Executes on selection change in pop17.
function pop17_Callback(hObject, eventdata, handles)
global op;
g4=['+' 'o' '*' 'x' 's' 'd'];
a=get(hObject,'Value');
op(6).m=g4(a);
hh=findall(gcf,'Tag','linia_mesura');
set(hh,'Marker',op(6).m);

% --- Executes during object creation, after setting all properties.
function pop17_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end
global op;
g4=['+' 'o' '*' 'x' 's' 'd'];
set(hObject,'Value',find(g4==op(6).m));

% --- Executes on selection change in pop18.
function pop18_Callback(hObject, eventdata, handles)
global op;
g5=[10 12 14 16 18 20];
a=get(hObject,'Value');
op(6).ms=g5(a);
hh=findall(gcf,'Tag','linia_mesura');
set(hh,'MarkerSize',op(6).ms);

% --- Executes during object creation, after setting all properties.
function pop18_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
global op;
g5=[10 12 14 16 18 20];
set(hObject,'Value',find(g5==op(6).ms));

% --- Executes on selection change in pop19.
function pop19_Callback(hObject, eventdata, handles)
global op;
g1=['r' 'g' 'b' 'c' 'm' 'y' 'k' 'w'];
a=get(hObject,'Value');
op(7).c=g1(a);
hh=findall(gcf,'Tag','text_mesura');
set(hh,'Color',op(7).c);

% --- Executes during object creation, after setting all properties.
function pop19_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
global op;
g1=['r' 'g' 'b' 'c' 'm' 'y' 'k' 'w'];
set(hObject,'Value',find(g1==op(7).c));

% --- Executes on selection change in pop20.
function pop20_Callback(hObject, eventdata, handles)
global op;
g2=[1 2 3 4 10 12 14 16 18 20];
a=get(hObject,'Value');
op(7).w=g2(a);
hh=findall(gcf,'Tag','text_mesura');
set(hh,'FontSize',op(7).w);

% --- Executes during object creation, after setting all properties.
function pop20_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
global op;
g2=[1 2 3 4 10 12 14 16 18 20];
set(hObject,'Value',find(g2==op(7).w));

% --- Executes on selection change in pop21.
function pop21_Callback(hObject, eventdata, handles)
global op;
g1=['r' 'g' 'b' 'c' 'm' 'y' 'k' 'w'];
a=get(hObject,'Value');
op(8).c=g1(a);

% --- Executes during object creation, after setting all properties.
function pop21_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in pop22.
function pop22_Callback(hObject, eventdata, handles)
global op;
g1=['r' 'g' 'b' 'c' 'm' 'y' 'k' 'w'];
a=get(hObject,'Value');
op(9).c=g1(a);
hh=findall(gcf,'Tag','linia_iso');
set(hh,'Color',op(9).c);

% --- Executes during object creation, after setting all properties.
function pop22_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
global op;
g1=['r' 'g' 'b' 'c' 'm' 'y' 'k' 'w'];
set(hObject,'Value',find(g1==op(9).c));

% --- Executes on selection change in pop23.
function pop23_Callback(hObject, eventdata, handles)
global op;
g2=[1 2 3 4 10 12 14 16 18 20];
a=get(hObject,'Value');
op(9).w=g2(a);
hh=findall(gcf,'Tag','linia_iso');
set(hh,'LineWidth',op(9).w);

% --- Executes during object creation, after setting all properties.
function pop23_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
global op;
```

```
g2=[1 2 3 4 10 12 14 16 18 20];
set(hObject,'Value',find(g2==op(9).w));

% --- Executes on selection change in pop24.
function pop24_Callback(hObject, eventdata, handles)
global op;
g3=['-' '--' ':' '-.'];
a=get(hObject,'Value');
op(9).s=g3(a);
if a==2
   op(9).s='--';
end
if a==4
   op(9).s='-.';
end
hh=findall(gcf,'Tag','linia_iso');
set(hh,'LineStyle',op(9).s);

% --- Executes during object creation, after setting all properties.
function pop24_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
global op;
g3=['-' '=' ':' '.'];
op2=op(9).s;
if op2=='--'
   op2='=';
end
if op2=='-.'
   op2='.';
end
set(hObject,'Value',find(g3==op2));

% --- Executes on selection change in pop25.
function pop25_Callback(hObject, eventdata, handles)
global op;
g4=['+' 'o' '*' 'x' 's' 'd'];
a=get(hObject,'Value');
op(9).m=g4(a);
hh=findall(gcf,'Tag','linia_iso');
set(hh,'Marker',op(9).m);

% --- Executes during object creation, after setting all properties.
function pop25_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
global op;
g4=['+' 'o' '*' 'x' 's' 'd'];
set(hObject,'Value',find(g4==op(9).m));

% --- Executes on selection change in pop26.
```

```
function pop26_Callback(hObject, eventdata, handles)
global op;
g5=[10 12 14 16 18 20];
a=get(hObject,'Value');
op(9).ms=g5(a);
hh=findall(gcf,'Tag','linia_iso');
set(hh,'MarkerSize',op(9).ms);

% --- Executes during object creation, after setting all properties.
function pop26_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
global op;
g5=[10 12 14 16 18 20];
set(hObject,'Value',find(g5==op(9).ms));

% --- Executes on selection change in pop27.
function pop27_Callback(hObject, eventdata, handles)
global op;
g1=['r' 'g' 'b' 'c' 'm' 'y' 'k' 'w'];
a=get(hObject,'Value');
op(10).c=g1(a);
hh=findall(gcf,'Tag','text_mesura_interior');
set(hh,'Color',op(10).c);

% --- Executes during object creation, after setting all properties.
function pop27_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
global op;
g1=['r' 'g' 'b' 'c' 'm' 'y' 'k' 'w'];
set(hObject,'Value',find(g1==op(10).c));

% --- Executes on selection change in pop28.
function pop28_Callback(hObject, eventdata, handles)
global op;
g2=[1 2 3 4 10 12 14 16 18 20];
a=get(hObject,'Value');
op(10).w=g2(a);
hh=findall(gcf,'Tag','text_mesura_interior');
set(hh,'FontSize',op(10).w);

% --- Executes during object creation, after setting all properties.
function pop28_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
global op;
g2=[1 2 3 4 10 12 14 16 18 20];
set(hObject,'Value',find(g2==op(10).w));
```

```
% --- Executes on selection change in pop29.
function pop29_Callback(hObject, eventdata, handles)
global op;
g1=['r' 'g' 'b' 'c' 'm' 'y' 'k' 'w'];
a=get(hObject,'Value');
op(11).c=g1(a);
hh=findall(gcf,'Tag','text_mesura_frontera');
set(hh,'Color',op(11).c);

% --- Executes during object creation, after setting all properties.
function pop29_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
global op;
g1=['r' 'g' 'b' 'c' 'm' 'y' 'k' 'w'];
set(hObject,'Value',find(g1==op(11).c));

% --- Executes on selection change in pop30.
function pop30_Callback(hObject, eventdata, handles)
global op;
g2=[1 2 3 4 10 12 14 16 18 20];
a=get(hObject,'Value');
op(11).w=g2(a);
hh=findall(gcf,'Tag','text_mesura_frontera');
set(hh,'FontSize',op(11).w);

% --- Executes during object creation, after setting all properties.
function pop30_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
global op;
g2=[1 2 3 4 10 12 14 16 18 20];
set(hObject,'Value',find(g2==op(11).w));

% --- Executes on selection change in pop31.
function pop31_Callback(hObject, eventdata, handles)
global op;
g1=['r' 'g' 'b' 'c' 'm' 'y' 'k' 'w'];
a=get(hObject,'Value');
op(31).c=g1(a);

% --- Executes during object creation, after setting all properties.
function pop31_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

% --------------------------------------------------------------------
function configuracio_pushbutton_tancar_Callback(hObject, eventdata, handles)
global conf1;
global conf2;
```

```
global conf3;
global conf4;
global conf5;
panels_invisibles(handles);
set(handles.calibrar_panel,'Visible',conf1);
set(handles.mida_gra_panel,'Visible',conf2);
set(handles.mida_gra_iso_panel,'Visible',conf3);
set(handles.duresa_panel,'Visible',conf4);
set(handles.quantificacio_panel,'Visible',conf5);
set(handles.configuracio_panel,'Visible','off');
set(handles.info_foto_panel,'Visible','on');

% --------------------------------------------------------------------
function editar_objecte_ClickedCallback(hObject, eventdata, handles)
global conf1;
global conf2;
global conf3;
global conf4;
global conf5;
conf1=get(handles.calibrar_panel,'Visible');
conf2=get(handles.mida_gra_panel,'Visible');
conf3=get(handles.mida_gra_iso_panel,'Visible');
conf4=get(handles.duresa_panel,'Visible');
conf5=get(handles.quantificacio_panel,'Visible');
panels_invisibles(handles);
set(handles.info_foto_panel,'Visible','on');
set(handles.configuracio_panel,'Visible','on');

% --------------------------------------------------------------------
function configuracio_pushbutton_desar_Callback(hObject, eventdata, handles)
global op;
save('opcions.mat','op');


% ************************************************************************
% ***** BOTÓ DE NIVELLS AUTOMÀTICS ***************************************
% ************************************************************************
% --------------------------------------------------------------------
function nivells_automatics_OnCallback(hObject, eventdata, handles)
global imatge;
global imatge_orig;
global imatge_o;
imatge_orig=imatge;
newfigure = figure('visible','off');
newaxes = copyobj(handles.axes1,newfigure);
delete(imhandles(newaxes))
axes(handles.axes1);
m=0.5;
for k=1:3
   t=0;
   im1=imatge(:,:,k);
   x=(0:255);
   h=hist(im1(:),x);
```

```
   t=cumsum(h.^m);
   t=t.*(256/max(t));
   imatge(:,:,k)=t(imatge(:,:,k)+1);
end
imatge=uint8(round(imatge));
hf1=imshow(imatge);
imatge_o=imatge;
axis image;
axis off;
copyobj(newaxes,gcf);
close(newfigure);

% --------------------------------------------------------------------
function nivells_automatics_OffCallback(hObject, eventdata, handles)
global imatge;
global imatge_orig;
newfigure = figure('visible','off');
newaxes = copyobj(handles.axes1,newfigure);
delete(imhandles(newaxes))
axes(handles.axes1);
imatge=imatge_orig;
imshow(imatge);
imatge_o=imatge;
axis image;
axis off;
copyobj(newaxes,gcf);
close(newfigure);


% ************************************************************************
% ***** BOTÓ DE MESURAR DISTÀNCIES ***************************************
% ************************************************************************
% La funció s'anomena mesura_linia_ClickedCallback
% La paraula ClickedCallback l'afegeix GUIDE, per indicar que es tracta
% d'una eina que està situada a la barra d'eines. S'activarà en el moment
% que es premi aquesta eina amb el ratolí.
% --------------------------------------------------------------------
function mesura_linia_ClickedCallback(hObject, eventdata, handles)
% Defineixen les variables escala i op com a globals, de manera que la
% funció pugui anar a buscar els seus valors a altres funcions.
global escala;
global op;
% Força Axes1 com els eixos actius (imatge principal)
axes(handles.axes1);
% Intenta executar la funció externa dibuixar_escala que, evidentment,
% dibuixa l'escala sobre la imatge principal. En cas que no pugui (perquè
% encara no s'hagi calibrat la imatge) executa les instruccions que hi han
% després de catch me (com que no n'hi ha cap, no farà res).
try
   dibuixar_escala();
catch me
end
% És una funció de MATLAB que permet que l'usuari dibuixi una línia de
```

```
% manera interactiva. A la línia dibuixada li assigna un handle, que el
% desa a la variable hml. El color de la línia vindrà especificat per les
% opcions del programa, desades en la matriu op.
hml=imline;
setColor(hml,op(5).c);
% El programa espera que l'usuari validi la línia fent un doble click al
% seu damunt, llegeix les seves coordenades, i les desa a la matriu linia.
coord = wait(hml);
linia=getPosition(hml);
% Esborra l'objecte hml. Com que hml és el handle o apuntador a la línia
% dibuixada, aleshores esborra la línia diguixada abans.
delete(hml);
% Dibuixa una nova línia a sobre justament de l'anterior, amb totes les
% característiques desades a la matriu d'opcions (color, gruix, estil de
% línia, estil del marcador final, mida d'aquest marcador i nom de la
% línia).
line([linia(1,1) linia(2,1)],[linia(1,2) linia(2,2)],...
   'Color',op(6).c,'LineWidth',op(6).w,'LineStyle',op(6).s,'Marker',op(6).m,...
   'MarkerSize',op(6).ms,'Tag','linia_mesura');
% Calcula la projecció horitzontal (Dx) i vertical (Dy) de la línia.
Dx=linia(2,1)-linia(1,1);
Dy=linia(2,2)-linia(1,2);
% Calcula l'angle que forma la línia respecte la horitzontal.
angle=(-1)*atan(Dy/Dx);
% Calcula la longitud de la línia per Pitàgores. Ho multiplica pel factor
% de conversió per passar de píxels a micres.
long=((Dx*Dx+Dy*Dy)^0.5)*escala;
% Calcula el punt central de la línia.
centrex=round((linia(1,1)+linia(2,1))/2);
centrey=round((linia(1,2)+linia(2,2))/2);
% Aplica una funció experimental per trobar la posició inicial del text de
% la mesura, a partir del centre de la línia i del seu angle
tx=150*sin(angle)-90;
ty=50*cos(angle);
% Escriu el text amb les dimensions de la línia amb totes les
% característiques desades a la matriu d'opcions (2 decimals, color, mida
% de lletra i nom de l'objecte).
text(centrex+tx,centrey+ty,sprintf('%0.2f µm',long),...
   'Color',op(7).c,'FontSize',op(7).w,'Tag','text_mesura');


% ************************************************************************
% ***** BOTÓ DE MESURAR SUPERFÍCIES ************************************
% ************************************************************************
% ------------------------------------------------------------------------
function mesura_superficie_ClickedCallback(hObject, eventdata, handles)
global escala;
global op;
hFH = imfreehand;
setColor(hFH,[0 1 0]);
binaryImage = hFH.createMask();
superficie=sum(sum(binaryImage==1))*escala*escala;
b=uint8(binaryImage);
```

```
l=graythresh(b);
b=im2bw(b,l);
m=regionprops(b, 'centroid');
centroids = double(cat(1, m.Centroid));
if size(centroids)==[0,0]
   centroids(1,1)=21;
   centroids(1,2)=22;
end
if centroids(1,1)<21
   centroids(1,1)=21;
end
if centroids(1,2)<22
   centroids(1,2)=22;
end
text(centroids(1,1)-70,centroids(1,2),sprintf('%0.0f µm²',superficie),...
   'Color',op(7).c,'FontSize',op(7).w,'Tag','text_mesura');


% ************************************************************************
% ***** BOTÓ DE COPIAR AL PORTAPAPERS ************************************
% ************************************************************************
% ------------------------------------------------------------------
function copiar_portapapers_ClickedCallback(hObject, eventdata, handles)
%
set(gcf,'InvertHardcopy','off');
print -dbitmap -noui;   %Exportar al portapapers
msgbox('Pantalla copiada al Portapapers!','Atenció','Help');


% ********************************** **********************************
% ***** BOTÓ D'EDITAR OBJECTES ***************************************
% ************************************************************************
% ------------------------------------------------------------------
function editor_objectes_OnCallback(hObject, eventdata, handles)
%
plotedit on;

% ------------------------------------------------------------------
function editor_objectes_OffCallback(hObject, eventdata, handles)
%
plotedit off;
```