Guía NDRVI Program

Información del documento	
Título del proyecto	NDRVI.01
Número de proyecto	00.00.01
Responsable del proyecto	AT
ID del entregable	Guía NDRVI Program
Edición	00.00.01

Contribuciones
HEMAV

Resumen

Guía de estructuración del programa NDRVI.

Participantes en la elaboración

Preparado por		
Nombre /Compañía	Posición/ Título	Fecha
Aitor Tena / HEMAV	Ingeniero	28/10/2013
…	…	…

Revisado Por		
Nombre /Compañía	Posición/ Título	Fecha
…	…	…

Aprobado por		
Nombre /Compañía	Posición/ Título	Fecha
…	…	…

Historial del documento

Edición	Fecha	Status	Autor	Justificación
00.00.01	28/10/2013	Nuevo	ATT	Nuevo documento

Contenido

Tabla de imágenes

1 Introducción

El presente documento pretende servir de guía dentro de la estructuración del programa NDRVI desarrollado por HEMAV en el área técnica de la agricultura de precisión.

La función principal del programa es el procesado de imágenes captadas en campos por las diferentes cámaras y equipos. Este procesado permite elaborar mapas con índices de vegetación para el estudio de los campos así como investigar y adquirir experiencia en el procesado.

El programa se articula en el entorno de MATLAB. Este entorno permite un desarrollo e implementación más fácil y rápido que otras plataformas por la gran variedad y cantidad de funciones que contiene. En la fecha de elaboración del documento se pretende seguir usando este lenguaje aunque no se descarta, en un futuro, poder implementarlo en lenguajes más avanzados y con más facilidad para la multiplataforma.

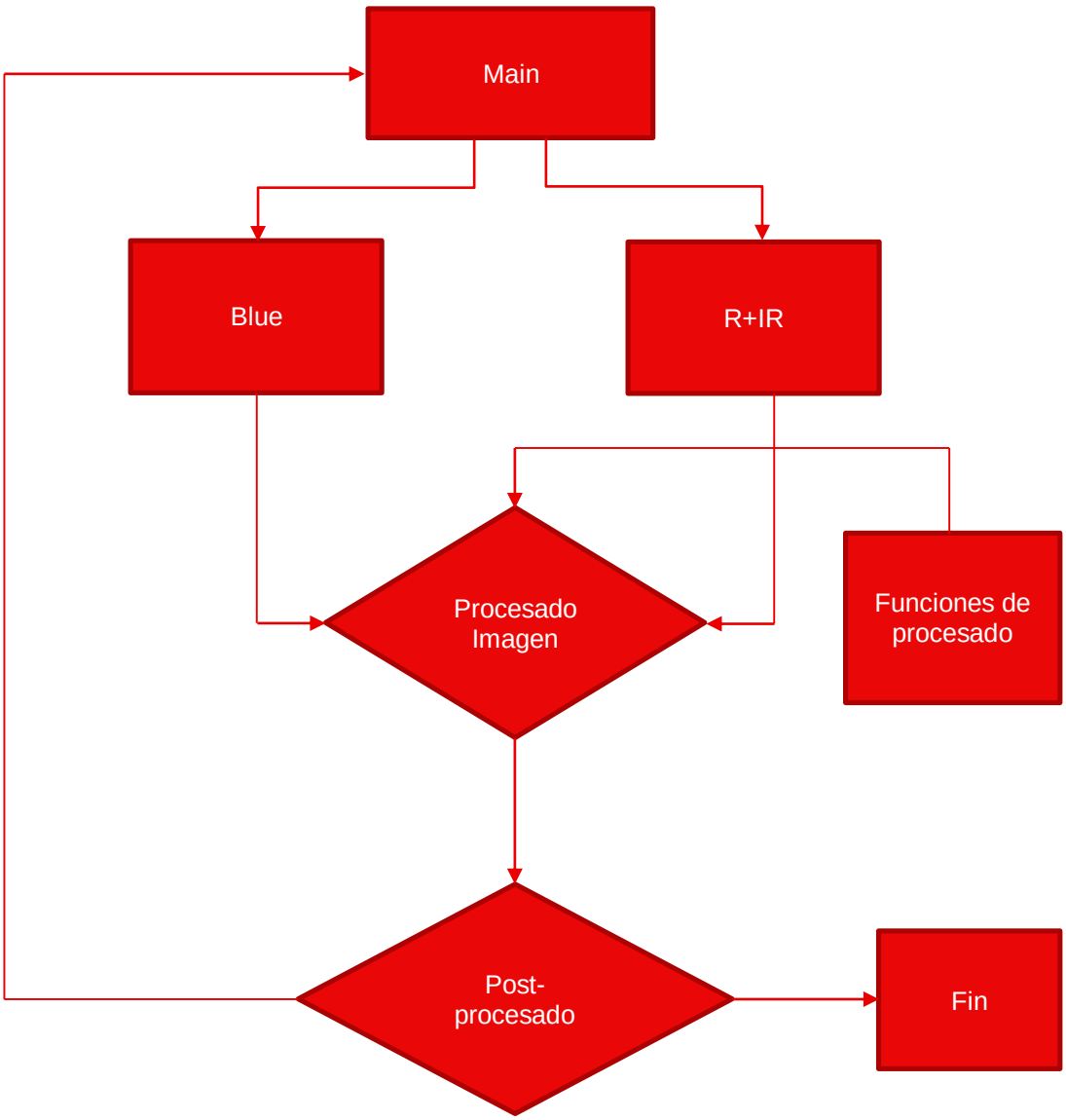
Este programa es propiedad de HEMAV y por lo tanto no se autoriza su distribución a ninguna persona/empresa, por el momento, que no sea del entorno HEMAV para trabajar con él o en su desarrollo.

2 Estructura del programa

Para facilitar la programación cuando se debe realizar un proyecto con tantos recursos el NDRVI se subdivide en diferentes archivos (*m-files*) cada uno con una función. La división del programa y sus secciones en diferentes archivos facilita la programación y la depuración de errores.

La idea es tener un archivo principal (interfaz del programa) en el cuál se llamen a las diferentes subsecciones para el procesado de las imágenes. A día de hoy el procesado se realiza en dos grandes bloques diferenciados por la técnica en el procesado de la imagen. Muchos de los módulos son compartidos por las diferentes secciones facilitando así la interoperabilidad entre las diferentes partes del programa.

La estructura general del programa es:



Normalmente cuando se modifica alguna función de procesado de imagen hay que modificar también alguna interfaz del menú y viceversa. Como se aprecia en el diagrama de bloques las funciones de procesado alimentan al procesado de imagen. De esta forma agregar más funciones no es ningún problema siempre y cuando se implementen bien en el procesado de la imagen.

Al final del procesado se puede acceder a un post-procesado de imagen con diferentes funciones para acabar de ajustar mejor las características de la imagen final.

Una vez se tiene la imagen final el programa da opción de volver al inicio o de finalizarlo. Esto es más cómodo para poder hacer diferentes pruebas con la misma imagen o al acabar de procesar una imagen continuar con la siguiente. Con el módulo *PostProc* se puede postprocesar la imagen para hacer un estudio de los terrenos más detallado.

Aunque los módulos de procesado *Blue* y *R+IR* compartan funciones éstos trabajan un poco diferente internamente. En la descripción de cada módulo se puede observar estas diferencias.

Además de los *m-files* el programa incorpora unas bases de datos en formato Excel guardadas en una carpeta aparte. Estas bases de datos sirven para agregar más opciones al programa sin tener que cambiar parte del código. Entre estas bases de datos se encuentra una librería la cual es una recopilación de las diferentes pruebas que se van llevando acabo al procesar diferentes imágenes.

La siguiente lista es una recopilación de todos archivos/carpetas del programa NDRVI (v4.0):

- NDRVI_Main (.m y .fig)
- NDRVI_Menu_Blue (.m y .fig)
- NDRVI_Menu_R_IR (.m y .fig)
- NDRVI_Im_PostProc (.m y .fig)
- LK_to_NDVI_Blue (.m)
- LK_to_NDVI_R_IR (.m)
- ND_BlueIR_to_LK (.m)
- ND_BlueB_to_LK (.m)
- ND_IR_to_LK (.m)
- ND_R_to_LK (.m)
- NDRVI_RGB_Menu (.m y .fig)
- NDRVI_Recalibration_IV (.m y .fig)
- NDRVI_im_save (.m)
- NDRVI_About (.m y .fig)
- Guía NDRVI Program (.pdf)
- HEMAV_logo (.jpg)
- DB_folder
    - BIBLIO_IND_db (.xlsx)
    - I_R_db (.xlsx)
    - IV_db (xlsx)

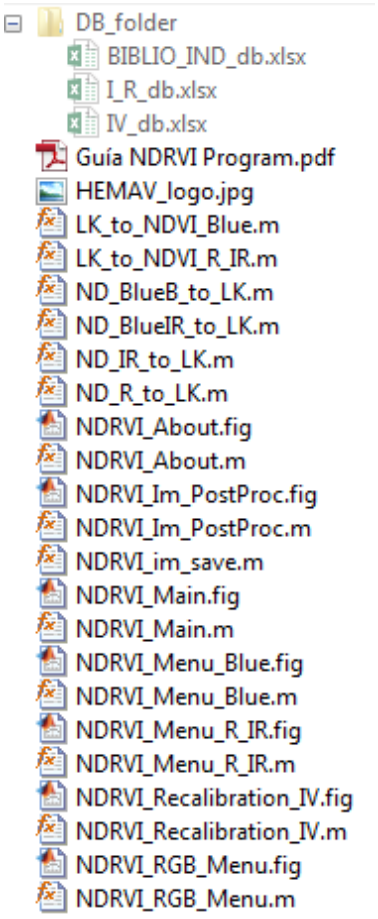


Imagen 1: Archivos programa

2.1 NDRVI_Main

La interfaz principal del programa permite seleccionar entre los dos módulos de procesamiento de imagen. El menú de la barra de tareas incorpora la opción de salir del programa en la pestaña *Archivo* y *Acerca de* en la pestaña *Ayuda*. La siguiente imagen muestra la interfaz en la versión 4.0:

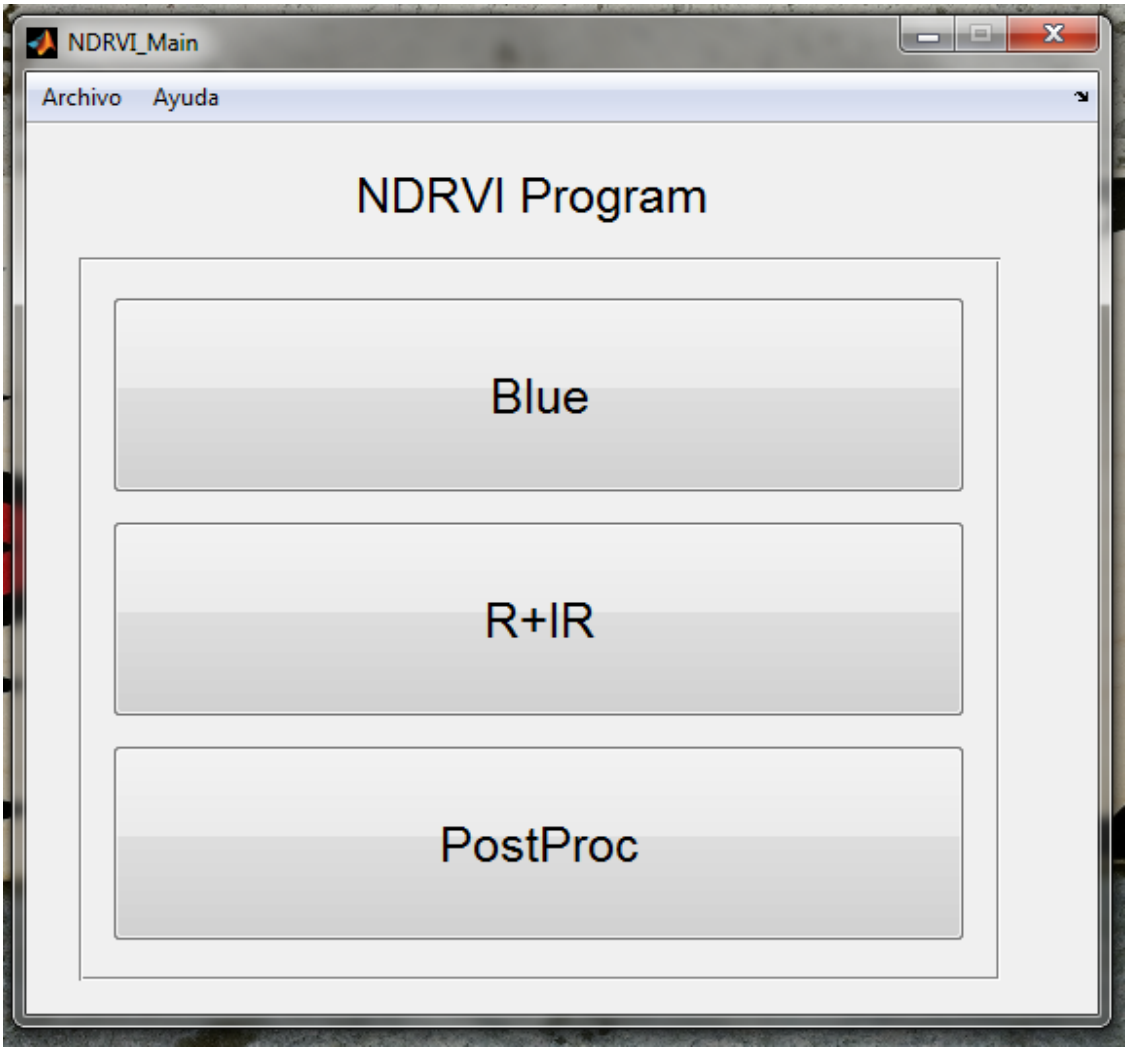


Imagen 2: Menú principal

Dentro del código está definida una variable *global* la cual es el nombre del programa que se muestra en la interfaz gráfica tanto del menú principal como de otras instancias:

```
global program_name;
program_name='NDRVI Program';
set(handles.title,'string',program_name);
```

Imagen 3: Variable *global* nombre programa

Una vez se inicializa la interfaz principal también lo hace esta variable y las otras partes del programa pueden utilizarla (como el programa está pensado para iniciarse siempre con la ventana principal no supone un conflicto para otras partes del programa). Esto supone que cambiar el nombre del programa solo hay que hacerlo modificando esta variable en esta parte del código (puesto que este nombre no es el definitivo).

La otra parte del código son funciones para llamar a los otros módulos. Cuando se llama a las funciones éstas cierran el *NDRVI_Main* e inicializan el módulo llamado, como ejemplo:

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

close NDRVI_Main;
NDRVI_Menu_R_IR;


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

close NDRVI_Main;
NDRVI_Menu_Blue;


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

close NDRVI_Main;
NDRVI_Im_PostProc;
```

Imagen 4: Ejemplo funciones en menú principal

2.2 NDRVI_About

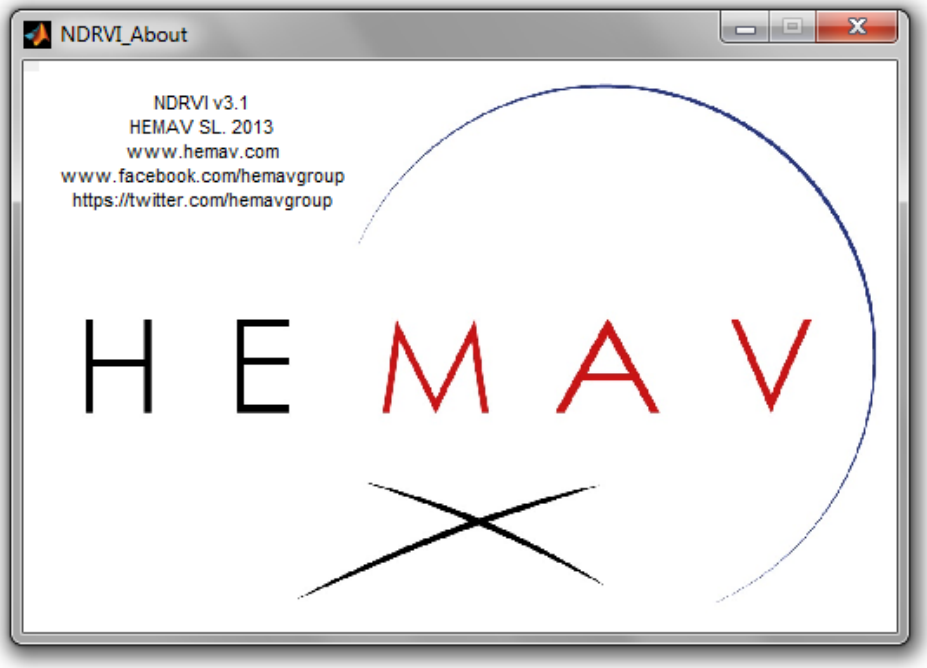


Imagen 5: Acerca de

La ventana de *Acerca de* sirve para informar sobre el programa y la empresa. Hay que actualizar el campo de texto de la versión cada vez que el programa cambie a una superior. Esta ventana se puede cerrar tradicionalmente con la cruz o clicando sobre ella en cualquier parte de la misma.

La imagen a mostrar se toma del archivo *HEMAV_logo.jpg* en la carpeta del programa. Ésta se muestra sobre unos *axes* con la función *imshow* de MATLAB. En el código hay una línea que permite cerrar la ventana con solo clicar encima:

```
axes(handles.axes1);
imshow('HEMAV_logo.jpg');
set(gcf,'WindowButtonDownFcn','close NDRVI_About')
```

Imagen 6: Código para cerrar ventana

2.3 NDRVI_Menu_Blue

En esta sección se describirá el módulo de procesamiento de imagen llamado *Blue* (por los filtros que se utilizan al tomar la foto) y las funciones internas. Para comenzar la interfaz del menú es la mostrada en la siguiente imagen (v4.0):

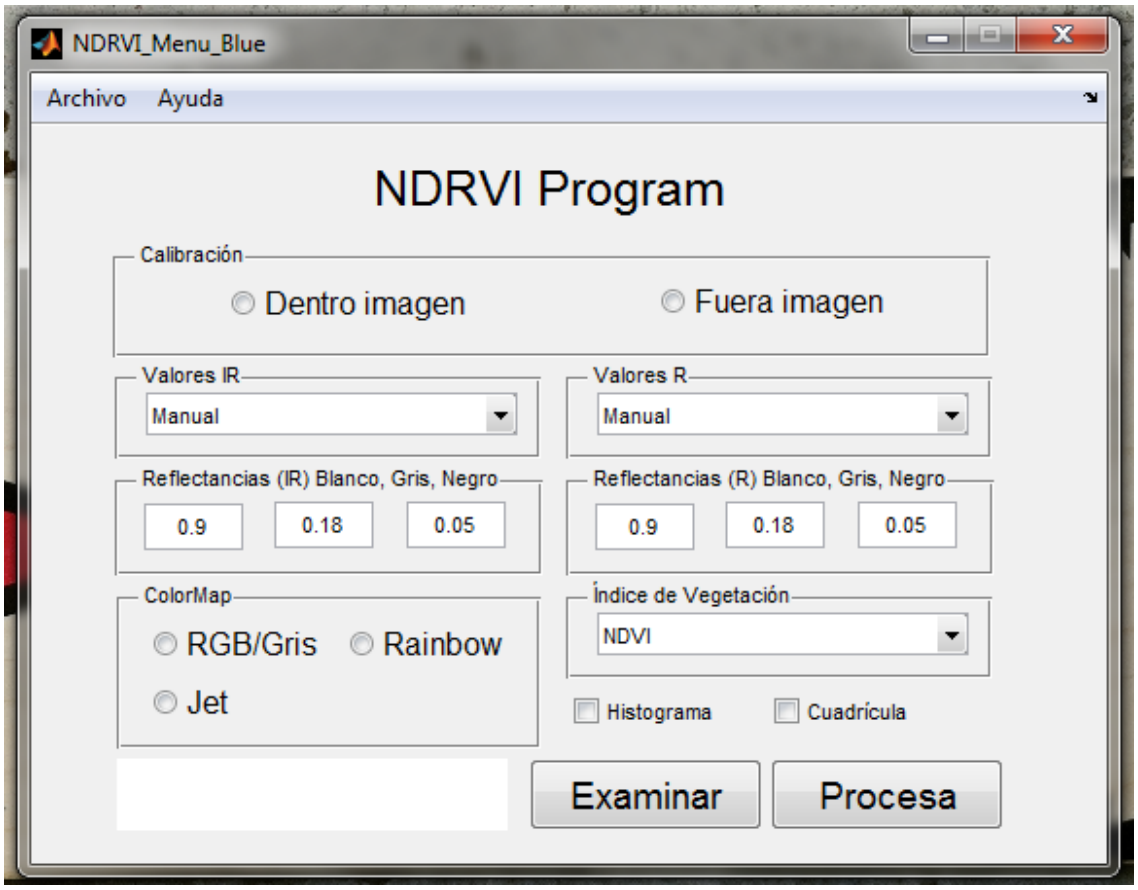


Imagen 7: Menú *Blue*

La interfaz está distribuida en varios sectores que ofrecen al usuario diferentes configuraciones para tratar la imagen. El usuario antes de apretar el botón *Procesa* debe escoger las opciones con las que se quiere procesar la imagen.

2.3.1 Inicio del menú Blue

En la inicialización de este menú se crean unas cuantas variables y se realizan algunas funciones antes de que la interfaz se muestre al usuario.

Primero se asigna el título del programa con la variable *global* vista en el menú principal. Al venir de ese menú esta variable ya está inicializada y no da problemas a la hora de asignar el nombre. Seguidamente se asigna el valor 0 a los *radiobuttons* de la parte de calibración. Con esto el usuario está obligado a escoger entre una de las dos opciones antes de procesar la imagen.

Ahora llega el turno de cargar valores de las bases de datos para mostrarlos en los *popupmenus* de los valores IR, valores R e índice de vegetación.

```
global program_name;
set(handles.title,'string',program_name);

set(handles.radiobutton_dentro_imagen,'value',0);
set(handles.radiobutton_fuera_imagen,'value',0);

[IR_db,IR_db_text]=xlsread('c\..\DB_folder\I_R_db.xlsx',1);
IR_db_size=size(IR_db);
str_list_IR{1}=IR_db_text{1};
for i=1:IR_db_size(1)
    str_list_IR{i+1}=strcat(num2str(IR_db(i,1)),'/',num2str(IR_db(i,2)));
end

[R_db,R_db_text]=xlsread('c\..\DB_folder\I_R_db.xlsx',2);
R_db_size=size(R_db);
str_list_R{1}=R_db_text{1};
for i=1:R_db_size(1)
    str_list_R{i+1}=strcat(num2str(R_db(i,1)),'/',num2str(R_db(i,2)));
end

str_list_IV=importdata('c\..\DB_folder\IV_db.xlsx');

set(handles.popupmenu_valores_IR,'string',str_list_IR);
set(handles.popupmenu_valores_R,'string',str_list_R);
set(handles.popupmenu_IV,'string',str_list_IV);

handles.IR_db=IR_db;
handles.R_db=R_db;
```

Imagen 8: Inicialización menu *Blue* 1

Para leer las bases de datos se utiliza la función *xlsread* puesto que por comodidad las bases de datos están guardadas en archivos de Excel. En el caso de los índices de vegetación se usa la función *importdata* ya que los campos son *string*. Después de leer los datos estos se cargan en los *popupmenus* correspondientes tal como muestra la imagen anterior. También se guardan los datos numéricos de las bases de datos en *handles.IR_db* y *handles.R_db* para su posterior uso en otras partes del código.

Para acabar se hace una limpieza de variables que no serán utilizadas y se asigna el valor 0 a los *radiobuttons* del panel de ColorMap. También se asignan unos valores por defecto a las reflectancias que pueden ser cambiados según convenga en esta parte del código o posteriormente por el usuario en la interfaz del programa.

```
clear IR_db IR_db_size IR_db_text R_db R_db_size R_db_text

set(handles.radiobutton_RGB_Gris,'value',0);
set(handles.radiobutton_Jet,'value',0);
set(handles.radiobutton_Rainbow,'value',0);

set(handles.edit_ref_blanco_IR,'String',num2str(0.9));
set(handles.edit_ref_gris_IR,'String',num2str(0.18));
set(handles.edit_ref_negro_IR,'String',num2str(0.05));
set(handles.edit_ref_blanco_R,'String',num2str(0.9));
set(handles.edit_ref_gris_R,'String',num2str(0.18));
set(handles.edit_ref_negro_R,'String',num2str(0.05));
% % %
```

Imagen 9: Inicialización menu *Blue* 2

2.3.2 Calibración

El procesamiento de la imagen para obtener los índices de vegetación requiere de un paso previo de calibración. En el panel de calibración debajo del título principal con el nombre del programa permite escoger entre una calibración dentro de la imagen o fuera de ésta.

Depende de la opción seleccionada dos variables dentro del programa toman valores 0 o 1. Estas dos variables son:

```
status_cal_dentro_imagen=get(handles.radiobutton_dentro_imagen,'Value');
status_cal_fuera_imagen=get(handles.radiobutton_fuera_imagen,'Value');
```

Imagen 10: Variables de calibración

Estas variables se empaquetan en un vector de calibración llamado *status_vector_calibracion* que será pasado a otra función.

Como se ve en la imagen anterior estas variables adquieren el valor del *radiobutton* ubicados en el panel de calibración para luego pasar estos valores a otras funciones.

En el caso de elegir la calibración fuera de la imagen en el momento de seleccionar el *radiobutton* aparecerá un nuevo menú de calibración. La parte del código es:

```
fuera_imagen_value=get(handles.radiobutton_fuera_imagen,'Value');
if fuera_imagen_value==1
    %Calibración IR
    [Im_Name,Im_PathName] = uigetfile({'*.jpg';'*.jpeg';'*.*'},'Selecciona imagen calibración');
    ND=double(imread(strcat(Im_PathName,Im_Name)));
    ND_IR=ND(:,:,1);
    minimo1=min(min(ND_IR));
    maximo1=max(max(ND_IR));
    ima1=figure;
    imshow(ND_IR);
    caxis([ minimo1 maximo1]);
    title('Marcar primero blanco, luego gris y finalmente negro (banda IR)');
    colorbar;
    [x,y]= ginput(3);
    maximo= ND_IR( round(y(1,1)) , round(x(1,1)) );  %blanco
    minimo_g= ND_IR( round(y(2,1)) , round(x(2,1)) );  %negro/gris
    minimo_n= ND_IR( round(y(3,1)) , round(x(3,1)) );  %negro/gris
    handles.x1_IR=[maximo minimo_g minimo_n];
    %Calibración R
    ND_IR  = ND(:,:,3);
    maximo= ND_IR( round(y(1,1)) , round(x(1,1)) );  %blanco
    minimo_g= ND_IR( round(y(2,1)) , round(x(2,1)) );  %negro/gris
    minimo_n= ND_IR( round(y(3,1)) , round(x(3,1)) );  %negro/gris
    handles.x1_R=[maximo minimo_g minimo_n];
    handles.x(1)=0; handles.x(2)=handles.x(1); handles.x(3)=handles.x(1);
    handles.y(1)=0; handles.y(2)=handles.y(1); handles.y(3)=handles.y(1);
    close(ima1);
    clear Im_PathName Im_Name ND ND_IR minimo1 maximo1 ima1 maximo minimo_g minimo_n
```

Imagen 11: Calibración fuera imagen

Las variables finales que interesan para la calibración son:

- *handles.x1_IR*
- *handles.x1_R*

por ello están definidas como *handle* para poder pasarlas a otras partes del código. En el caso de no seleccionar la calibración fuera de la imagen estas variables toman el valor 0:

```
else
    handles.x1_IR(1)=0; handles.x1_IR(2)=handles.x1_IR(1); handles.x1_IR(3)=handles.x1_IR(1);
    handles.x1_R(1)=0; handles.x1_R(2)=handles.x1_R(1); handles.x1_R(3)=handles.x1_R(1);
    handles.x(1)=0; handles.x(2)=handles.x(1); handles.x(3)=handles.x(1);
    handles.y(1)=0; handles.y(2)=handles.y(1); handles.y(3)=handles.y(1);
end
```

Imagen 12: Variables calibración dentro imagen valor 0

2.3.3 Valores IR R

Los valores IR y R se utilizan en una parte del procesamiento de la imagen para interpolar. En los *popupmenus* se pueden seleccionar unos valores por defecto o manualmente. En el caso de la selección manual cuando se procese la imagen el usuario podrá introducir estos valores. Si se selecciona cualquier valor por defecto un par de variable adquirirán cierto valor (1, 2, 3…) en función de la opción escogida. Estas variables son:

```
status_valores_IR=get(handles.popupmenu_valores_IR,'Value');
status_valores_R=get(handles.popupmenu_valores_R,'Value');
```

Imagen 13: Valores IR R

Los *popupmenus* devuelven el valor de la posición escogida dentro de la lista desplegable (1,2 3…). En el Excel de las bases de datos los valores están ordenados de forma que coinciden con la numeración de estos menús. El valor 1 sirve para que el programa sepa que el usuario introducirá manualmente los valores más adelante así pues la posición real del valor para que corresponda con el del *popupmenu* estará desplazada una unidad.

2.3.4 Reflectancias IR R

Como se mencionó anteriormente estos valores se asignan en el inicio del código y se muestran en sus correspondientes casillas en los paneles de las reflectancias. Los valores por defecto se pueden cambiar dentro del código o el usuario puede elegir valores diferentes una vez se abre el menú.

Los valores de las reflectancias se pasan a otras funciones con las variables (vectores) siguientes:

```
vector_reflectancias_IR(1)=str2double(get(handles.edit_ref_blanco_IR,'String'));
vector_reflectancias_IR(2)=str2double(get(handles.edit_ref_gris_IR,'String'));
vector_reflectancias_IR(3)=str2double(get(handles.edit_ref_negro_IR,'String'));
vector_reflectancias_R(1)=str2double(get(handles.edit_ref_blanco_R,'String'));
vector_reflectancias_R(2)=str2double(get(handles.edit_ref_gris_R,'String'));
vector_reflectancias_R(3)=str2double(get(handles.edit_ref_negro_R,'String'));
```

Imagen 14: Variables reflectancias

2.3.5 ColorMap

El *colormap* es un atributo para mostrar imágenes en MATLAB que sirve para colorear con diferentes mapas de color la imagen en cuestión. En el panel de selección del *colormap* el usuario puede elegir entre tres opciones:

- RGB/Gris
- Jet
- Rainbow

Se definen unas variables para saber que opción ha escogido el usuario:

```
status_cmap_rgb_g=get(handles.radiobutton_RGB_Gris,'Value');
status_cmap_jet=get(handles.radiobutton_Jet,'Value');
status_cmap_rb=get(handles.radiobutton_Rainbow,'Value');
```

Imagen 15: Variables *colormap* 1

Como en el caso de la calibración, estas variables se empaquetan en un vector llamado *status_vector_cmap*.

Si el usuario elige el *colormap* RGB/Gris inmediatamente se llama a una función (NDRVI_RGB_Menu) para seleccionar los límites de los colores asociados. Esta función se detalla más adelante. Cuando se inicializa el *NDRVI_Menu_Blue* se crean las variables de los

límites (tanto RGB como las bandas blancas auxiliares) y se les da el valor 0. También se inicializa la variable *handles.status_suelo* y *handles.check_aux* a 0:

```
handles.rgb_g_limits(1)=0; handles.rgb_g_limits(2)=handles.rgb_g_limits(1);
handles.rgb_g_limits(3)=handles.rgb_g_limits(1); handles.rgb_g_limits(4)=handles.rgb_g_limits(1);
handles.rgb_g_limits(5)=handles.rgb_g_limits(1); handles.rgb_g_limits(6)=handles.rgb_g_limits(1);

handles.status_suelo=0;

handles.auxiliar_limits(1)=0; handles.auxiliar_limits(2)=handles.auxiliar_limits(1);
handles.auxiliar_limits(3)=handles.auxiliar_limits(1); handles.auxiliar_limits(4)=handles.auxiliar_limits(1);
handles.auxiliar_limits(5)=handles.auxiliar_limits(1); handles.auxiliar_limits(6)=handles.auxiliar_limits(1);

handles.check_aux(1)=0; handles.check_aux(2)=handles.check_aux(1); handles.check_aux(3)=handles.check_aux(1);
```

Imagen 16: Variables *colormap* 2

2.3.6 Índice de Vegetación

En el panel de índice de vegetación el usuario puede escoger entre los diferentes métodos para procesar la imagen para obtener los índices de vegetación. Los nombres de los índices se cargan des del Excel con la base de datos y se pueden añadir más en caso que sea necesario sin tener que modificar el código.

La variable *status_NDVI* recoge el valor del *popupmenu* asociado a este panel y pasa el valor para que las funciones sepan que método usar.

2.3.7 Opción histograma

El usuario puede elegir que una vez finalizado el procesado de la imagen se cargue en una nueva ventana un histograma correspondiente a la imagen procesada. De este modo es más fácil analizar la imagen y poder discriminar valores.

La variable *status_hist* recoge el valor del *checkbox* asociado a esta opción para pasarlo a otras funciones. Los valores que puede tomar son 0 o 1 dependiendo de si está seleccionada o no la casilla.

Antes de mostrar el histograma el código crea una variable auxiliar donde copia la imagen procesada y la recorre buscando todos los valores negativos y si encuentra estos valores los cambia a 0.

2.3.8 Opción cuadrícula

Esta opción permite al usuario dibujar una parrilla encima de la imagen. Una ventana aparece delante del usuario y éste elige cuantas divisiones horizontales y verticales quiere pudiendo así dividir la imagen en sectores para un mejor análisis.

Las variables *status_cuad* y *handles.cuad_div* se pasan a otras funciones para indicar si la opción de cuadrícula está activa y en ese caso cuantas divisiones quiere el usuario. De modo contrario si no se elige la opción cuadrícula las variables *status_cuad* y *handles.cuad_div* toman valor 0.

2.3.9 Botón Examinar

Este botón permite al usuario elegir la imagen a procesar. También asigna la ruta completa y nombre del archivo a un recuadro de texto para que el usuario pueda identificarlo mejor.

```
[Im_Name,Im_PathName] = uigetfile({'*.jpg';'*.jpeg';'*.*'},'Selecciona imagen procesado');
handles.full_img_path_name=strcat(Im_PathName,Im_Name);
set(handles.text_examinar,'String',handles.full_img_path_name);
clear Im_Name Im_PathName
guidata(hObject, handles);
```

Imagen 17: Botón examiner

La ruta completa del archivo elegido se guarda en la variable *handles.full_img_path_name* para pasarla posteriormente a otras funciones.

2.3.10 Botón Procesa

Como se ha ido viendo en las otras secciones cada parte de este código recopila las opciones que elige el usuario para procesar la imagen y las prepara para pasar estas variable a otra función (la encargada de procesar la imagen).

La función a la cual se llama es *LK_to_NDVI_Blue* y se le pasan los siguientes argumentos:

- *status_valores_IR*
- *status_valores_R*
- *handles.IR_db*
- *handles.R_db*
- *status_vector_calibracion*
- *handles.x1_IR*
- *handles.x1_R*
- *vector_reflectancias_IR*
- *vector_reflectancias_R*
- *status_vector_cmap*
- *status_hist*
- *status_cuad*
- *handles.cuad_div*
- *handles.rgb_g_limits*
- *handles.auxiliar_limits*
- *handles.status_suelo*
- *handles.check_aux*
- *status_NDVI*
- *handles.full_img_oath_name*

2.4 NDRVI_Menu_R_IR

El menú de procesado *NDRVI_Menu_R_IR* es muy similar al anterior, solo sufre algunas modificaciones por su diferente mecanismo para procesar las imágenes por ello solo se explicarán sus particulares diferencias.

Para este módulo se necesitan dos imágenes tomadas con diferentes cámaras. La dificultad radica en que tienen que ser imágenes tomadas des de la misma posición y que sean lo más parecidas posibles. Una imagen se toma en visible y la otra con filtro.

El menú tiene el aspecto siguiente:

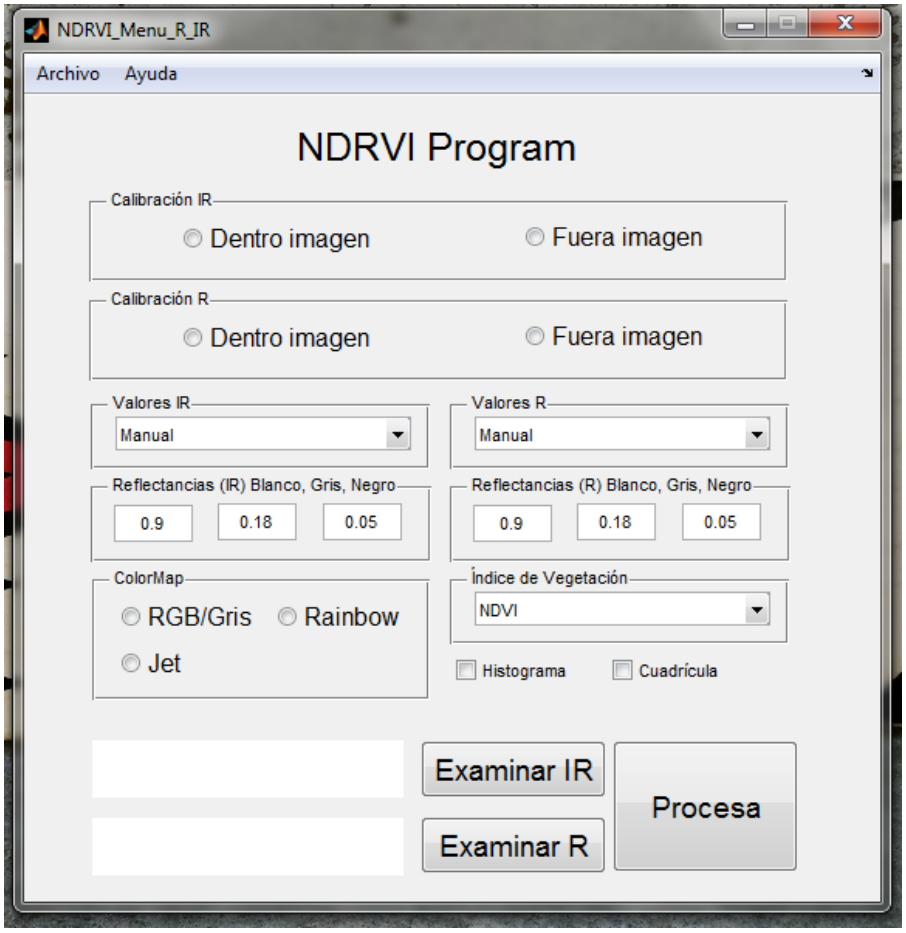


Imagen 18: Menú *R_IR*

Se aprecia en la imagen anterior que aparecen tanto dos opciones de calibración como dos selecciones de imágenes. Gracias a esto se puede calibrar cada imagen (visible o filtro) tanto dentro de ellas como fuera o cualquiera de las otras 3 combinaciones. Como se calibran dos imágenes habrá que pasar el doble de argumentos de calibración

Los argumentos que se pasan a la función *LK_to_NDVI_R_IR* son:

- *status_valores_IR*
- *status_valores_R*
- *handles.IR_db*
- *handles.R_db*
- *status_vector_calibracion_IR*
- *status_vector_calibracion_R*
- *handles.x1_IR*

- *handles.x1_R*
- *vector_reflectancias_IR*
- *vector_reflectancias_R*
- *status_vector_cmap*
- *status_hist*
- *status_cuad*
- *handles.cuad_div*
- *handles.rgb_g_limits*
- *handles.auxiliar_limits*
- *handles.status_suelo*
- *handles.check_aux*
- *status_NDVI*
- *handles.full_img_path_name_IR*
- *handles.full_img_path_name_R*

2.5 NDRVI_Im_PostProc

El módulo *NDRVI_Im_PostProc* está pensado para postprocesar una imagen después de haberla tratado con cualquiera de los dos módulos *Blue* o *R+IR*.

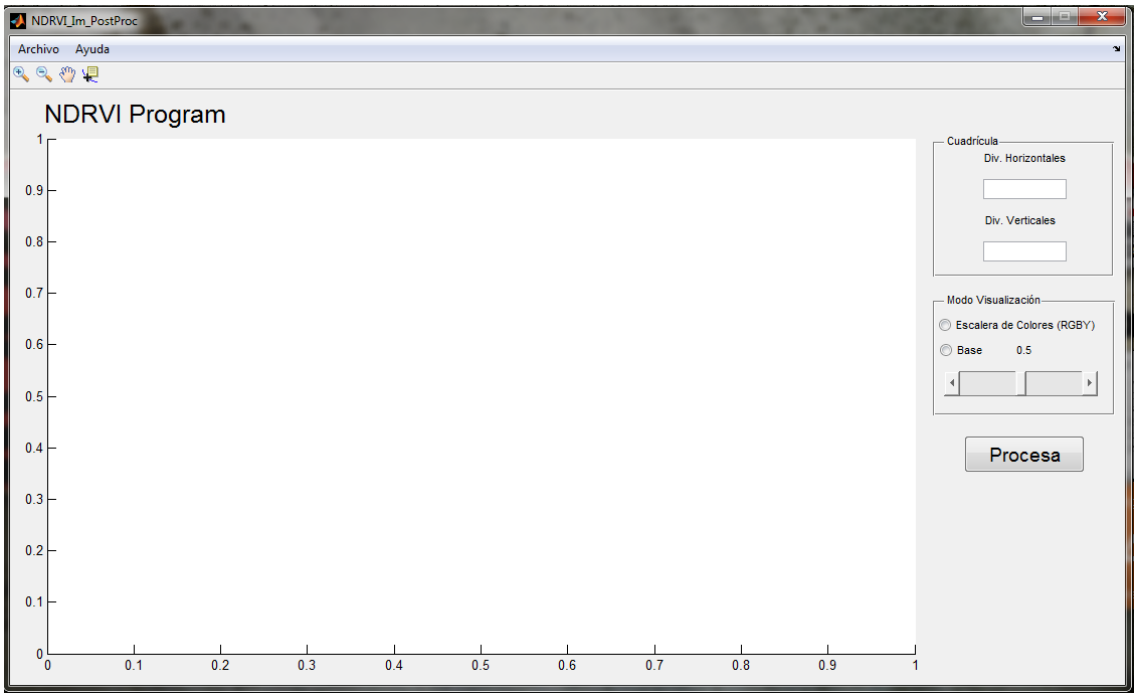


Imagen 19: Módulo *NDRVI_Im_PostProc*

Dispone de un menú desplegable arriba a la izquierda para cargar una imagen, guardar una imagen procesada, guardar la figura de una imagen procesada, volver al menú principal o cerrar el módulo definitivamente.

Una vez cargada la imagen se pueden añadir las divisiones que se quiera (verticales y horizontales) para crear la cuadrícula. Esta cuadrícula ayuda a procesar partes de la imagen haciendo medias de los valores del índice de vegetación ayudando así al análisis de la imagen.

Empezando por el inicio de la función en el código se encuentra:

```
%%%
global program_name;
set(handles.title,'string',program_name);
set(handles.radiobutton_RGBY,'value',0);
set(handles.radiobutton_base,'value',0);
set(handles.slider_base,'Min',0);
set(handles.slider_base,'Max',1);
set(handles.slider_base,'Value',0.5);
set(handles.text_base_slider,'String',num2str(get(handles.slider_base,'Value')));
handles.start_x=0;
handles.start_y=0;
%%%
```

Imagen 20: Inicio *NDRV_Im_PostProc*

Como siempre el nombre del programa seguido de la asignación de los diferentes elementos de control (*radiobuttons*, *sliders*, etc) sus respectivos valores iniciales. Las variables *handles.start_x* y *handles.start_y* se les asigna un 0 al principio para que el código sepa que es la primera vez que se utilizan y luego pueda recordar los valores de las divisiones. Esto se verá más claro cuando se llegue a la parte del código que las necesita.

La parte de código que carga la imagen también se encarga de convertirla a *double* y de normalizarla para poder trabajar con ella (dividir todos los valores entre 255). Al llevar su tiempo se muestra una barra de carga para que el usuario sepa lo que está sucediendo.

```
[Im_Name,Im_PathName] = uigetfile({'*.jpg';'*.jpeg';'*.*'},
full_img_path_name=strcat(Im_PathName,Im_Name);
handles.ima=double(imread(full_img_path_name));
handles.im_size=size(handles.ima);

current_process=waitbar(0,'Cargando imagen...0%');
for i=1:handles.im_size(1)
    waitbar(i/handles.im_size(1),current_process,strcat('Ca
    for j=1:handles.im_size(2)
        handles.ima(i,j)=handles.ima(i,j)/255;
    end
end
close(current_process);

axes(handles.axes1);
imshow(handles.ima);
colorbar

guidata(hObject, handles);
```

Imagen 21: Carga imagen postprocesado

Una vez se introducen las divisiones se puede escoger que *colormap* visualizar. Si no se escoge *colormap* el módulo no lo aplicará al mostrar la imagen por lo tanto ésta se verá como inicialmente se cargó. Al apretar el botón de *PROCESA* el código que se ejecuta es el siguiente:

```
ima_mitja=handles.ima;
cuad_div_x=str2double(get(handles.edit_div_horiz,'String'));
cuad_div_y=str2double(get(handles.edit_div_ver,'String'));
status_mod_vis(1)=get(handles.radiobutton_RGBY,'Value');
status_mod_vis(2)=get(handles.radiobutton_base,'Value');
mitja_sum=0;
pixels=0;
```

Imagen 22: Postprocesado 1

Primero se crea una copia de la imagen ya que sus valores se alteraran y si el usuario decide hacer otro postprocesado no hace falta que recargue la imagen. Seguidamente se adquieren los valores de las divisiones y que *colormap* se ha escogido. La variable *mitja_sum* se posiciona en 0 puesto que será la encargada de guardar las medias de cada sector. La variable *pixels* también se le asigna el valor cero ya que contará cuantos pixels se procesan por sector.

```
% Crear media
current_process=waitbar(0,'Procesando imagen...0%');
for x_recor=1:cuad_div_x
    waitbar(x_recor/cuad_div_x,current_process,strcat('Procesando imagen...',num2str(round(x_recor*100/cuad_div_x)),'%'));
    for y_recor=1:cuad_div_y
        for i=(x_recor-1)*floor(handles.im_size(2)/cuad_div_x)+1:x_recor*floor(handles.im_size(2)/cuad_div_x)
            for j=(y_recor-1)*floor(handles.im_size(1)/cuad_div_y)+1:y_recor*floor(handles.im_size(1)/cuad_div_y)
                mitja_sum=mitja_sum+handles.ima(j,i); pixels=pixels+1;
            end
        end
        mitja=mitja_sum/pixels;
        mitja_sum=0;
        pixels=0;
        for i=(x_recor-1)*floor(handles.im_size(2)/cuad_div_x)+1:x_recor*floor(handles.im_size(2)/cuad_div_x)
            for j=(y_recor-1)*floor(handles.im_size(1)/cuad_div_y)+1:y_recor*floor(handles.im_size(1)/cuad_div_y)
                ima_mitja(j,i)=mitja;
            end
        end
    end
end
close(current_process);
```

Imagen 23: Medias sectores postprocesado

Llegados a este punto el código empieza a recorrer la imagen por sectores y calculando las medias de cada uno. Básicamente esta parte del código recorre un sector, calcula la media del índice de vegetación y recorre otra vez el sector asignando el valor medio a cada pixel. En este proceso también se crea una barra de progreso para informar al usuario. Hay que añadir que este código puede contener ciertos *bugs* que debieran ser corregidos más adelante:

- No está limitado el número de divisiones pudiendo sobrepasar los píxeles mismos.
- Las divisiones entre sectores pueden dar problemas ya que hay que hacer redondeos, sobre todo en los bordes derecho e inferior de la imagen.

```
if status_mod_vis(1)==1 && status_mod_vis(2)==0
    rgby_cmap=zeros(4,3);
    rgby_cmap(1,1)=1; rgby_cmap(1,2)=0; rgby_cmap(1,3)=0;
    rgby_cmap(2,1)=0; rgby_cmap(2,2)=1; rgby_cmap(2,3)=0;
    rgby_cmap(3,1)=0; rgby_cmap(3,2)=0; rgby_cmap(3,3)=1;
    rgby_cmap(4,1)=1; rgby_cmap(4,2)=1; rgby_cmap(4,3)=0;
    colormap(rgby_cmap)
end

if status_mod_vis(1)==0 && status_mod_vis(2)==1
    base=floor(100*get(handles.slider_base,'Value'));
    base_cmap=zeros(100,3);
    for i=base+1:100
        base_cmap(i,1)=1; base_cmap(i,2)=1; base_cmap(i,3)=1;
    end
    colormap(base_cmap)
end

colorbar
```

Imagen 24: *Colormap* postprocesado

La imagen anterior muestra el cálculo de los *colormaps* según se haya escogido uno u otro. Finalmente se grafica la cuadrícula con las mismas líneas de código vistas anteriormente.

El menú *Archivo* se puede escoger guardar la imagen procesada según que *colormap* se haya escogido (o aunque no se haya escogido ninguno) con la función *NDRVI_im_save*. El guardado de la imagen no guarda la cuadrícula (se debe implementar en un futuro como opción a escoger). También se puede guardar la figura de la imagen procesada con su respectivo *colormap* y *colorbar*.

2.6 NDRVI_RGB_Menu

El propósito de este menú es establecer los límites para el *colormap* de las bandas RGB/Gris así como dejar o quitar el suelo (de la fotografía que se procese). Además ofrece la posibilidad de añadir tres bandas blancas auxiliares para ayudar a discriminar mejor. La interfaz del menú ofrece unas casillas en las cuales delimitar los límites de las bandas. La particularidad de estas casillas es que al establecer el límite superior de la banda Azul automáticamente se establece el límite inferior de la banda Verde con el mismo valor. Lo mismo pasa con el límite superior de la banda Verde y el límite inferior de la banda Roja.

Imagen 25: Menú selección RGB

Debajo de las casillas existe el botón *Quitar suelo* el cual se puede marcar o no para procesar la fotografía con o sin suelo. La función de este es pintar el suelo de blanco. Para ello todos los valores que estén por debajo del límite inferior de la banda Azul se pintarán de blanco si la casilla de *Quitar suelo* está activa.

El usuario puede activar las bandas auxiliares con las *checkbox* de la derecha y luego rellenando los límites. Este menú también tiene memoria, una vez procesada la imagen si se vuelve a abrir recuerda los valores y las opciones seleccionadas así se hace más cómodo la discriminación en el procesado.

Por último el botón *OK* cierra el menú y pasa los valores al *NDRVI_Menu_Blue*. Para realizar este pase de argumentos se utiliza un vector llamado *handles.rtrn* que pasa los 7 argumentos al menú principal.

```
handles.rtrn(1)=str2double(get(handles.edit_Banda_Azul,'String'));
handles.rtrn(2)=str2double(get(handles.edit_hasta_Azul,'String'));
handles.rtrn(3)=str2double(get(handles.edit_Banda_Verde,'String'));
handles.rtrn(4)=str2double(get(handles.edit_hasta_Verde,'String'));
handles.rtrn(5)=str2double(get(handles.edit_Banda_Roja,'String'));
handles.rtrn(6)=str2double(get(handles.edit_hasta_Roja,'String'));
handles.rtrn(7)=get(handles.checkbox_suelo,'Value');

guidata(hObject, handles);

close NDRVI_RGB_Menu;
```

Imagen 26: Vector retorno argumentos bandas

La siguiente imagen muestra la función de salida que pasa los argumentos al menú principal:

```
%varargout{1} = handles.output;
varargout{1} = handles.rtrn(1); %Banda Azul
varargout{2} = handles.rtrn(2); %Hasta Banda Azul
varargout{3} = handles.rtrn(3); %Banda Verde
varargout{4} = handles.rtrn(4); %Hasta Banda Verde
varargout{5} = handles.rtrn(5); %Banda Roja
varargout{6} = handles.rtrn(6); %Hasta Banda Roja
varargout{7} = handles.rtrn(7); %Suelo
varargout{8} = handles.rtrn(8); %Banda Auxiliar 1
varargout{9} = handles.rtrn(9); %Hasta Banda Auxiliar 1
varargout{10} = handles.rtrn(10); %Banda Auxiliar 2
varargout{11} = handles.rtrn(11); %Hasta Banda Auxiliar 2
varargout{12} = handles.rtrn(12); %Banda Auxiliar 3
varargout{13} = handles.rtrn(13); %Hasta Banda Auxiliar 3
varargout{14} = handles.rtrn(14); %Check Auxiliar 1
varargout{15} = handles.rtrn(15); %Check Auxiliar 2
varargout{16} = handles.rtrn(16); %Check Auxiliar 3
% The figure can be deleted now
delete(handles.figure1);
```

Imagen 27: Función salida menu RGB/Gris

2.7 LK_to_NDVI_Blue

Esta función es la parte central del procesamiento de las imágenes. Una vez se recopilan todos los datos de entrada de *NDRVI_Menu_Blue* y se le pasan a esta función ya puede empezar el procesado de las fotografías.

Durante el procesado aparece una barra de proceso para indicar al usuario que acciones está haciendo el programa. Esta barra de proceso introduce unos retardos para darle tiempo al usuario de leer la información en pantalla.

```
current_process=waitbar(0,'Calculando valores de irradiancia IR');
pause on; pause(1.5); pause off;
```

Imagen 28: Ejemplo barra proceso con retardo

La función va llamando otras subfunciones las cuales procesan la imagen en las bandas necesarias para obtener los índices de vegetación correspondientes. Las subfunciones a las cuales llama son la *ND_BlueIR_to_LK* y *ND_BlueB_to_LK*:

```
current_process=waitbar(0,'Calculando valores de irradiancia IR');
pause on; pause(1.5); pause off;
close(current_process);
[ND_IR,x,y]  = ND_BlueIR_to_LK (status_valores_IR,IR_db,status_vector_calibracion,x1_IR,x,y,vector_reflectancias_IR,full_img_path_name);
current_process=waitbar(0.33,'Calculando valores de irradiancia IR');
pause on; pause(1.5); pause off;
close(current_process);
current_process=waitbar(0.33,'Calculando valores de irradiancia R');
pause on; pause(1.5); pause off;
close(current_process);
ND_RED = ND_BlueB_to_LK (status_valores_R,R_db,status_vector_calibracion,x1_R,x,y,vector_reflectancias_R,full_img_path_name);
current_process=waitbar(0.66,'Calculando valores de irradiancia R');
pause on; pause(1.5); pause off;
close(current_process);
```

Imagen 29: Llamada a las funciones *BlueIR* y *BlueB*

Estas subfunciones devuelven unas matrices con las imágenes procesadas. Seguidamente se calculan los índices de vegetación entre las diferentes opciones:

```
switch status_NDVI

    case 1

        ima=(ND_IR-ND_RED)./(ND_IR+ND_RED);

    case 2

        ima=((ND_IR-ND_RED)./(ND_IR+ND_RED+0.5))*1.5; %SAVI

    case 3

        ima=(((ND_IR-ND_RED)./(ND_IR+ND_RED))+1)/2; %%NDVI infragram

    case 4

        ima=(ND_IR)./(ND_RED); %PaD

    case 5

        ima=(((ND_IR)./(ND_RED))+1)/2; %PaD Infragram

end
```

Imagen 30: Cálculo índices de vegetación

Este cálculo suele devolver algunos valores que se salen del rango numérico (NaN). Por ello a continuación se vuelve a procesar la matriz de la imagen para eliminar estos valores y dejarlos en -1:

```
%Eliminación NAN=-1
s_ima=size(ima);
for i=1:s_ima(1)
    for j=1:s_ima(2)
        if isnan(ima(i,j))==1
            ima(i,j)=-1;
        end
    end
end
```

Imagen 31: Eliminación de los NaN

A continuación se aplica el *colormap* seleccionado en el menú principal. Para el caso del RGB/Gris se construye el *colormap* en base a los límites escogidos por el usuario y si hay suelo o no. Para ello este *colormap* está basado directamente en el que trae por defecto MATLAB (*gray*) pero modificado para introducir los colores de las bandas en los límites correspondientes. El *colormap Jet* se usa directamente de MATLAB. Para el *colormap Rainbow* se construye directamente si se selecciona la opción en el menú principal. Para ahorrar tiempo y recursos de cálculo en una futura versión se podría guardar un archivo con la matriz del *colormap Rainbow* que se cargase en caso necesario.

```
case 3 %Rainbow

    rainbow=zeros(10,3);

    rainbow(1,1)=1; rainbow(1,2)=0; rainbow(1,3)=0;
    rainbow(2,1)=1; rainbow(2,2)=127/255; rainbow(2,3)=0;
    rainbow(3,1)=1; rainbow(3,2)=1; rainbow(3,3)=0;
    rainbow(4,1)=0; rainbow(4,2)=1; rainbow(4,3)=0;
    rainbow(5,1)=0; rainbow(5,2)=1; rainbow(5,3)=127/255;
    rainbow(6,1)=0; rainbow(6,2)=1; rainbow(6,3)=1;
    rainbow(7,1)=0; rainbow(7,2)=127/255; rainbow(7,3)=1;
    rainbow(8,1)=0; rainbow(8,2)=0; rainbow(8,3)=1;
    rainbow(9,1)=127/255; rainbow(9,2)=0; rainbow(9,3)=1;
    rainbow(10,1)=1; rainbow(10,2)=0; rainbow(10,3)=1;

    cmap=rainbow;
    colormap(rainbow);
```

Imagen 32: *Colormap Rainbow*

Además, cuando se selecciona el *colormap* éste se guarda también en una variable *cmap* para pasarla posteriormente a otra subfunción.

Después según si se ha seleccionado o no la opción del histograma el programa abrirá o no la ventana del histograma correspondiente a la imagen. También se mostrará en una ventana los valores máximo y mínimo de los índices de vegetación.

Posteriormente se le preguntará al usuario si quiere recalibrar los índices de vegetación. Si selecciona que sí se abrirá una nueva ventana para tal efecto (el recalibrado de los índices de vegetación se comentará más adelante).

```
if status_hist==1
    figure;
    imhist(ima);
end

max_min=msgbox(strcat('Máximo: ',num2str(maximo),' Mínimo: ',num2str(minimo)),'Valor Máximo y Mínimo');
waitfor(max_min);
choice=questdlg('Recalibrar índices?','Recalibración Índices de Vegetación','Sí','No','Sí');

if strcmp(choice,'Sí')==1

    NDRVI_Recalibration_IV(0,ima,cmap,maximo,minimo);

else
```

Imagen 33: Histograma, máximo, mínimo y recalibrado

Por último el código guardará la figura y la imagen procesada. El tema de guardar la imagen procesada con su respectivo *colormap* no está solucionado aun (v3.1).

2.8 LK_to_NDVI_R_IR

Esta función es prácticamente igual que la anterior, la única diferencia son los argumentos de entrada debido a los diferentes procesados por lo demás es idéntica.

Cualquier cambio efectuado en la función anterior ha de hacerse en esta y viceversa.

2.9 ND_BlueIR_to_LK

Entre las diferentes partes del código para procesar la imagen cabe destacar las relacionadas con las opciones seleccionables del menú principal. La primera es la calibración de la imagen dentro o fuera de ella. Si se seleccionó dentro, esta es la parte del código que permite esta opción:

```
if status_vector_calibracion(1)==1 && status_vector_calibracion(2)==0

    minimo1 = min(min(ND_IR));
    maximo1 = max(max(ND_IR));
    ima1=figure;
    imshow(ND_IR);
    caxis([ minimo1 maximo1]);
    title('Marcar primero blanco, luego gris y finalmente negro (banda IR)');
    colorbar;
    [x,y]= ginput(3);
    maximo= ND_IR( round(y(1,1)) , round(x(1,1)) );  %blanco
    minimo_g= ND_IR( round(y(2,1)) , round(x(2,1)) );  %negro/gris
    minimo_n= ND_IR( round(y(3,1)) , round(x(3,1)) );  %negro/gris
    x1_IR=[maximo minimo_g minimo_n];
    close(ima1);

end
```

Imagen 34: Código calibración dentro imagen

Si no, el programa continúa con los valores que se le han pasado des del menú principal.

Por otro lado está la opción de introducir manualmente los índices de reflectancias y los parámetros de interpolación:

```
if status_valores_IR==1

    prompt={'Reflectancia blanco:','Reflectancia gris:','Reflectancia negro:','Param. interpolación 1:','Param. interpolación 2:'};
    dlg_title='Parámetros';
    num_lines=1;
    def={'0.9','0.18','0.05','200','150'};
    answer=inputdlg(prompt,dlg_title,num_lines,def);
    data=str2double(answer);

    y1(1)=data(1); y1(2)=data(2); y1(3)=data(3); %y1=[0.9 0.18 0.1];
    xi(1)=data(4); xi(2)=data(5); %xi=[220 180];
    yi=interp1(x1_IR,y1,xi,'linear');

else

    y1(1)=vector_reflectancias_IR(1); y1(2)=vector_reflectancias_IR(2); y1(3)=vector_reflectancias_IR(3); %y1=[0.9 0.18 0.1];
    xi(1)=IR_db(status_valores_IR-1,1); xi(2)=IR_db(status_valores_IR-1,2); %xi=[220 180];
    yi=interp1(x1_IR,y1,xi,'linear');

end
```

Imagen 35: Índices reflectancia y parámetros interpolación manual

Las otras partes del código son para el procesado de la imagen y en un principio no deben ser modificadas (hasta que sea necesario por una optimización o nuevo método de cálculo).

Los datos de salida de esta subfunción son la imagen procesada en su correspondiente banda y dos vectores de coordenadas *x* e *y*. Estos dos últimos vectores se le pasaran a la subfunción *ND_BlueB_to_LK* para la calibración dentro de la imagen. En caso de haber hecho esta calibración fuera de la imagen estos vectores son 0 y no se utilizarán.

2.10 ND_BlueB_to_LK

Como en la subfunción anterior (*ND_BlueIR_to_LK*) ésta procesa la imagen con un código muy similar. La diferencia está en la banda que coge para realizar los cálculos y que en caso de utilizar la calibración dentro de la imagen los vectores *x* e *y* que se pasaron de la subfunción anterior se utilizaran automáticamente para la calibración sin necesidad de que el usuario escoja otra vez los puntos. Este hecho es así para evitar coger puntos diferentes entre bandas cuando se procesan, consiguiendo así mejores resultados.

```
if status_vector_calibracion(1)==1 && status_vector_calibracion(2)==0

    minimo1 = min(min(ND_IR));
    maximo1 = max(max(ND_IR));
    ima1=figure;
    imshow(ND_IR);
    caxis([ minimo1 maximo1]);
    colorbar;
    maximo= ND_IR( round(y(1,1)) , round(x(1,1)) );  %blanco
    minimo_g= ND_IR( round(y(2,1)) , round(x(2,1)) );  %negro/gris
    minimo_n= ND_IR( round(y(3,1)) , round(x(3,1)) );  %negro/gris
    x1_R=[maximo minimo_g minimo_n];
    close(ima1);

end

if status_valores_R==1

    prompt={'Reflectancia blanco:','Reflectancia gris:','Reflectancia n
    dlg_title='Parámetros';
    num_lines=1;
    def={'0.9','0.18','0.05','200','100'};
    answer=inputdlg(prompt,dlg_title,num_lines,def);
    data=str2double(answer);

    y1(1)=data(1); y1(2)=data(2); y1(3)=data(3); %y1=[0.9 0.18 0.1];
    xi(1)=data(4); xi(2)=data(5); %xi=[220 180];
    yi=interp1(x1_R,y1,xi,'linear');

else
```

Imagen 36: Parte código *ND_BlueB_to_LK*

2.11 ND_IR_to_LK

2.12 ND_R_to_LK

2.13 NDRVI_im_save

El objetivo de esta función es guardar la imagen procesada con su *colormap* y sin él para disponer de las imágenes en buena calidad.

Para ello primero de todo se comprueban todos los valores de los píxeles y si están por debajo de 0 se les asigna este valor:

```
if min(min(ima))<0
    for j=1:im_size(2)
        for i=1:im_size(1)
            if ima(i,j)<0
                ima(i,j)=0;
            end
        end
    end
end
```

Imagen 37: Asignación valor 0 guardar imagen

Después se crea una diálogo para que el usuario pueda introducir el nombre de la imagen y la calidad de guardado. El formato por defecto es *jpg* aunque se podría incluir una opción para seleccionar también la extensión de la imagen.

Seguidamente se hace un recorrido por toda la imagen para asignar a cada valor del pixel su correspondiente color del *colormap*. Este proceso debería poder realizarlo MATLAB con la función *imwrite* pero por causas desconocidas no es capaz de hacerlo. Una barra de progeso muestra al usuario el avance.

```
current_process=waitbar(0,'Guardando imagen...0%');

for j=1:im_size(2)
    waitbar(j/im_size(2),current_process,strcat('Guardando imagen...',num2str(round(j*100/im_size(2))),'%'));
    for i=1:im_size(1)
        k=2;
        while ima(i,j)>k/im_cmap(1)
            k=k+1;
        end
        if abs(ima(i,j)-k/im_cmap(1))<abs(ima(i,j)-(k-1)/im_cmap(1))
            im_save(i,j,1)=cmap(k,1); im_save(i,j,2)=cmap(k,2); im_save(i,j,3)=cmap(k,3);
        else
            im_save(i,j,1)=cmap(k-1,1); im_save(i,j,2)=cmap(k-1,2); im_save(i,j,3)=cmap(k-1,3);
        end
    end
end

close(current_process);
```

Imagen 38: Recorrido asignación *colormap*

Por último se guarda la imagen tanto con *colormap* y sin él. La imagen con *colormap* llevará en su nombre pegado *cmap* para distinguir entre las dos.

2.14 NDRVI_Recalibration_IV

Al iniciar este módulo se llama a la función *NDRVI_Recalibration_IV* a la cual se le pasan los argumentos: *0*, *ima*, *cmap*, *maximo*, *minimo*. El primero es para establecer el valor del índice de recalibrado a 0. El segundo es la imagen a recalibrar seguido del *colormap* utilizado para la representación de la imagen. Por último están los valores máximo y mínimo del índice de vegetación de la imagen. Estos argumentos se entran en la función mediante *varargin* y se asignan a otras variables *handles* para mayor facilidad a la hora de escribir.

```
% % %
global program_name;
set(handles.title,'string',program_name);

%varargin{1} = primer índice de calibración seteado a 0
%varargin{2} = imagen "ima"
%varargin{3} = colormap
%varargin{4} = maximo
%varargin{5} = minimo

handles.IV_rec=varargin{1};
handles.ima=varargin{2};
handles.cmap=varargin{3};
handles.maximo=varargin{4};
handles.minimo=varargin{5};

handles.size_ima=size(handles.ima);
handles.delta_IV=0.01;

set(handles.edit_delta_IV,'String',num2str(varargin{1}));
```

Imagen 39: Inicio código recalibrado

Otra parte importante del código de inicio es la carga de la imagen en los *axes*. Este proceso está hecho de tal forma que haciendo clic derecho con el ratón encima de la imagen se despliegue un menú contextual con diferentes opciones:

```
axes(handles.axes1);
handles.im=imagesc(handles.ima);
colormap(handles.cmap)
set(handles.axes1,'xtick',[]);
set(handles.axes1,'xcolor',[0.941 0.941 0.941])
set(handles.axes1,'ytick',[]);
set(handles.axes1,'ycolor',[0.941 0.941 0.941])
set(handles.im,'HitTest','off')
caxis([0,1])
colorbar;
```

Imagen 40: Carga imagen *axes*

La imagen se ajustará a la forma de los *axes* así que es posible que dependiendo de la imagen a mostrar se distorsione un poco al ser mostrada en pantalla.

El módulo de recalibrado sirve para una vez procesada la fotografía poder variar los valores del índice de vegetación de la imagen. Para ello en la interfaz se dispone de un recuadro en el cual el usuario puede escribir el valor incremental que quiera:

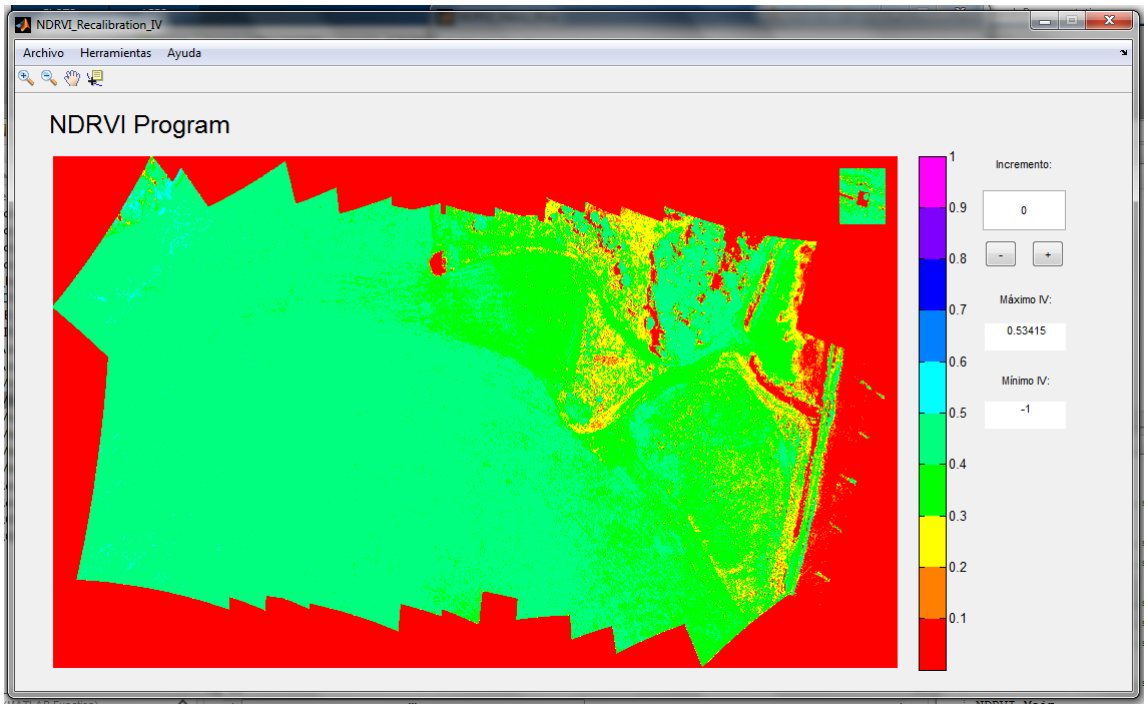


Imagen 41: Interfaz recalibrado

Además, debajo del recuadro incremental también existen dos botones con las designaciones + o – para realizar incrementos de 0.01. Estos incrementos se pueden cambiar fácilmente en el código ya que están asociados a la variable *handles.delta_IV* definida al inicio del código.

Debajo aparecen dos recuadros más no modificables indicando el valor máximo y el valor mínimo del índice de vegetación en la imagen procesada. Cuando se realiza un incremento, ya sea positivo o negativo, el valor máximo del índice de vegetación nunca superará la unidad (1) y el valor mínimo nunca estará por debajo de -1.

Al hacer un incremento aparece una barra de proceso que indica que se está llevando a cabo el recalibrado. Este proceso puede llevar un tiempo ya que el programa recorre toda la matriz de la imagen aplicando el incremento. La parte del código es la siguiente:

```
handles.IV_rec=str2double(get(handles.edit_delta_IV,'String'));
current_process=waitbar(0,'Recalibrando...0%');
for i=1:handles.size_ima(1)
    waitbar(i/handles.size_ima(1),current_process,strcat('Recalibrando...',num2str(round(i*100/handles.size_ima(1))),'%'));
    for j=1:handles.size_ima(2)
        if handles.ima(i,j)+handles.IV_rec>=1
            handles.ima(i,j)=1;
        end
        if handles.ima(i,j)+handles.IV_rec<=-1
            handles.ima(i,j)=-1;
        end
        if handles.ima(i,j)+handles.IV_rec<1 && handles.ima(i,j)+handles.IV_rec>-1
            handles.ima(i,j)=handles.ima(i,j)+handles.IV_rec;
        end
    end
end
close(current_process);
```

Imagen 42: Ejemplo código recalibrado

Este código se repite tanto para un incremento positivo, incremento negativo y para el recuadro incremental.

Dentro de la venta existe un menú de herramientas que consta de: *Archivo*, *Herramientas* y *Ayuda*. El menú *Archivo* da la opción de guardar la imagen recalibrada o cerrar la ventana para volver al programa principal. El menú de *Herramientas* consta de la opción de crear un histograma a partir de la imagen recalibrada. Esta misma opción está disponible en un menú contextual que se despliega al hacer clic derecho con el ratón encima de la imagen. Por último está el menú *Ayuda* que lleva a la ventana de *Acerca de*.

3 Otras partes del programa

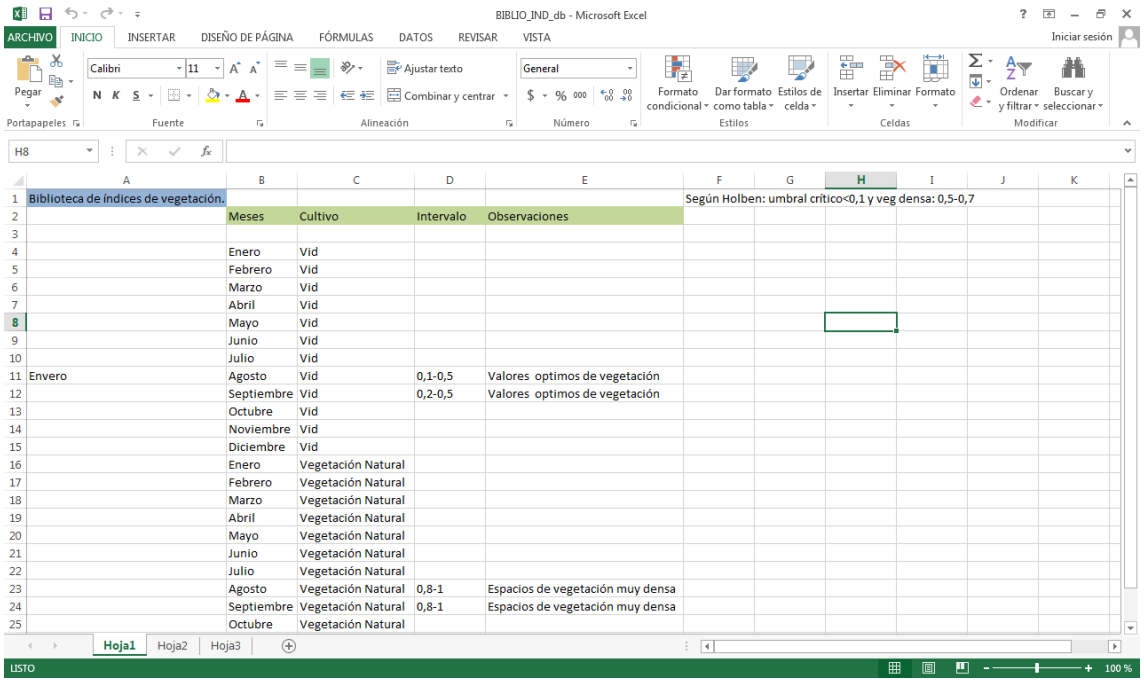
En esta sección se describen otras partes del programa igualmente importantes para el funcionamiento del mismo.

3.1 Bases de datos

Para facilitar la recolección de datos y la introducción de estos en el código se han creado unas bases de datos que se cargan en diferentes módulos del programa.

3.1.1 Biblioteca de índices

Esta biblioteca se basa en un libro Excel donde se recogen pruebas y datos para su posterior estudio y/o consulta. Es una manera de guardar pruebas ensayo/error y tener una base donde consultar.



	A	B	C	D	E	F
1	Biblioteca de índices de vegetación.					Según Holben: umbral crítico<0,1 y veg densa: 0,5-0,7
2		Meses	Cultivo	Intervalo	Observaciones	
3						
4		Enero	Vid			
5		Febrero	Vid			
6		Marzo	Vid			
7		Abril	Vid			
8		Mayo	Vid			
9		Junio	Vid			
10		Julio	Vid			
11	Envero	Agosto	Vid	0,1-0,5	Valores optimos de vegetación	
12		Septiembre	Vid	0,2-0,5	Valores optimos de vegetación	
13		Octubre	Vid			
14		Noviembre	Vid			
15		Diciembre	Vid			
16		Enero	Vegetación Natural			
17		Febrero	Vegetación Natural			
18		Marzo	Vegetación Natural			
19		Abril	Vegetación Natural			
20		Mayo	Vegetación Natural			
21		Junio	Vegetación Natural			
22		Julio	Vegetación Natural			
23		Agosto	Vegetación Natural	0,8-1	Espacios de vegetación muy densa	
24		Septiembre	Vegetación Natural	0,8-1	Espacios de vegetación muy densa	
25		Octubre	Vegetación Natural			

Imagen 43: Biblioteca índices

Con el nombre *BIBLIO_IND_db* este archivo se guarda en la carpeta *DB_folder*. Para abrir esta biblioteca dentro del programa se puede hacer en los menús principales de procesado en la pestaña *Archivo* opción *Biblioteca*. La función que carga el libro Excel es:

```
% --------------------------------------------------------------------
function menu_editor_Archivo_Biblioteca_Callback(hObject, eventdata, handles)
% hObject    handle to menu_editor_Archivo_Biblioteca (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

winopen('c\..\DB_folder\BIBLIO_IND_db.xlsx');
```

Imagen 44: Función carga biblioteca índices

Cuando se carga des del programa se abre el libro Excel y se puede modificar y guardar sin ningún problema. Se recomienda que cada nueva versión del programa compilada para poder incluir la biblioteca actualizada se guarde una copia en un archivo externo des del programa y se utilice para compilar en la nueva versión.

3.1.2 Base de datos de interpolación

Una parte del código de procesado de la imagen interpola entre ciertos valores. Según convenga es bueno poder variar los límites de interpolación o disponer de unas opciones rápidas para el procesado. Por ello se guardan valores para el procesado de IR o R en un libro Excel. Este libro se estructura de la misma manera, dos columnas encabezadas por la palabra *Manual* seguidas por valores numéricos. Dentro del libro existen dos hojas cada una para los valores IR y R respectivamente.

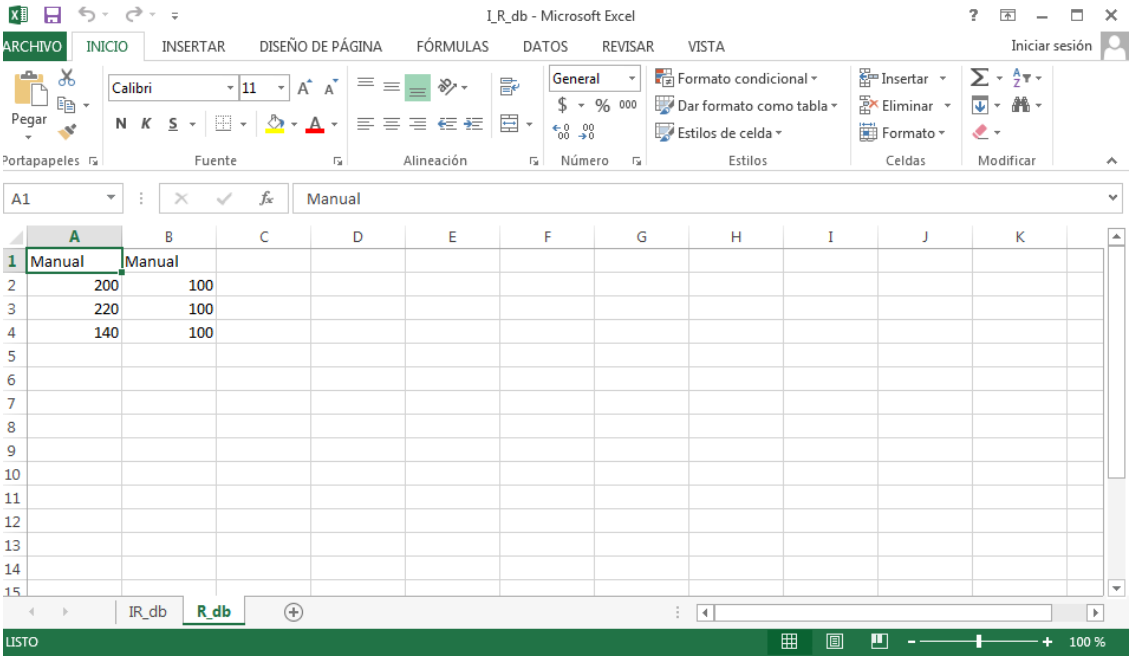


Imagen 45: Libro *I_R_db*

El código carga el libro (*I_R_db*) con la función *xlsread* que separa la parte de caracteres de la numérica y permite seleccionar la hoja a leer. Los valores se pasan a unas matrices para ser utilizados en los *popupmenus*. En caso de querer añadir más opciones solo es necesario modificar este Excel añadiendo más valores en las columnas, el código se encargará de cargarlos y disponerlos en las listas.

3.1.3 Base de datos de índices de vegetación

La manera de calcular los índices de vegetación no es única, según la discriminación que se quiera hacer, tipo de vegetación, etc. existen diferentes fórmulas. El programa permite escoger y añadir maneras de calcular los índices de vegetación.

El libro Excel *IV_db* contiene los nombres en columna de los diferentes métodos de cálculo. Para añadir más solo hay que poner el nombre del nuevo método a continuación del último y el programa se encargará de mostrarlo en el *popupmenu*. A su vez hay que añadir la fórmula del nuevo método en la parte del código correspondiente al procesado de la imagen.

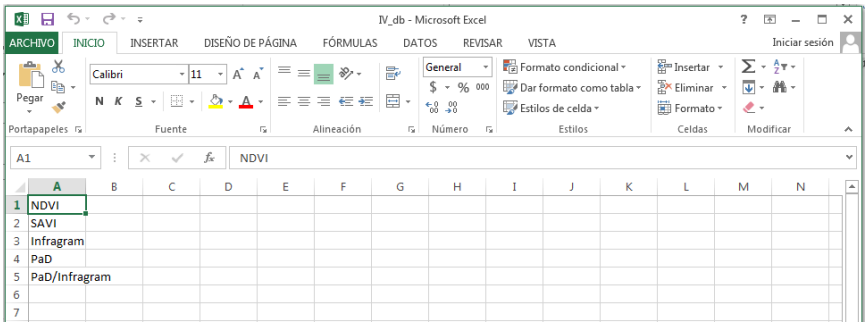


Imagen 46: Base de datos *IV_db*

Esta base de datos se carga con la función *importdata* de MATLAB en una variable que se pasa al *popupmenu* correspondiente.

```
str_list_IV=importdata('c\..\DB_folder\IV_db.xlsx');
```

Imagen 47: Función carga base de datos *IV_db*

3.1.4 Guía NDRVI Program

En el menú principal del programa (*NDRVI_Main*) está la opción de consultar la guía del programa en formato *pdf* en la pestaña *Ayuda* y luego *Guía*. Al seleccionar esta opción el programa abre la guía (se requiere tener un programa de visualizado de documentos *pdf*).

A medida que se actualice la guía del programa se deberá guardar también en formato *pdf* en la misma carpeta que las funciones del programa con el nombre *Guía NDRVI Program* (a no ser que se cambie el nombre del programa).

```
% --------------------------------------------------------------------
function menu_editor_Ayuda_Guia_Callback(hObject, eventdata, handles)
% hObject    handle to menu_editor_Ayuda_Guia (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

winopen('Guía NDRVI Program.pdf');
```

Imagen 48: Función carga guía programa

4 Cambios y actualizaciones

En esta sección se recogen los cambios y/o actualizaciones relevantes de una versión a otra del programa (a partir de la versión v3.1).

4.1 v3.1 y anteriores

Hasta esta versión el programa es capaz de/incluye:

- Interfaz gráfica para casi todos los procesos.
- Procesar imágenes en el módulo *Blue*.
- Guardar imágenes procesadas (formato baja calidad).
- Aplicar diferentes fórmulas de cálculo de índices de vegetación.
- Calibrar la imagen dentro de ella o fuera con otra imagen.
- Aplicar tres tipos diferentes de *colormaps*: RGB/Gris, Jet y Rainbow.
- Introducir valor de reflectancias antes del procesado.
- Opción para introducir valores de procesado manualmente.
- Cargar datos de ficheros externos para llenar los *popupmenus*.
- Recalibrar índices después de procesar la imagen.
- Barras de progeso.
- Crear histogramas a partir de las imágenes.
- Incorpora biblioteca de índices.
- Incorpora barra de menús y menús contextuales.

4.2 v3.2

- Añadida guía del programa.
- Corrección de pequeños errores.

4.3 v4.0

- Corrección errores código.
- Implementación módulo R+IR.
- Implementación módulo de post procesado de imagen.
- Opción cuadrícula.
- Guardado imagen con colormap con calidad.
- Bandas auxiliares (3 bandas blancas).
- Guardado imagen/figura en el módulo de post procesado de imagen.
- Valores medios máximo y mínimo en el módulo de post procesado de imagen.
- Actualizada guía del programa.