Scripting imlook4d

Table of Contents

Introduction

Imlook4d is tightly integrated with the Matlab workspace. This allows export from imlook4d to the workspace, and import back from the workspace. Thus, following an export the image matrix and other variables may be manipulated from the Matlab command window, thus unleashing the full potential of Matlab. The data can then be imported back again to imlook4d.

This very principle of calculating in the Matlab workspace, is exactly what is done when running a Matlab script file from the command line. Imlook4d facilitates the use of scripts, by integrating scripts. Inside the main folder where imlook4d was installed, there is a folder named SCRIPTS. Imlook4d scans the subfolders of the SCRIPTS folder for files with the extention ".m".

Basics – the mechanism for scripting

When imlook4d starts, the SCRIPTS folder and one subfolder down is analyzed for .m files. The SCRIPTS menu is populated by the subfolder names. Each of these subfolder names becomes a menu item in the SCRIPTS menu. Within each of these menu items (the subfolder names), new submenu items are created with names from the script names.

Naming convention: Since Matlab script names are not allowed to include spaces, but it is nice for readibility to have spaces in the menues, the convention to use "_" in the script name when a space is intended should be used. imlook4d will replace "_" with " " in the menu items.

Clicking the SCRIPTS menu exports the variables `imlook4d_current_handle`, `imlook4d_current_handles` to Matlab workspace. The first variable is a handle to the active imlook4d window (also called an imlook4d instance). The second variable is a structure containing all data within this instance.

Following the click on the SCRIPTS menu it unfolds, and a submenu item is selected. When a submenu item is selected the script file with the same name is executed (compare comment about spaces above).

Basics – create an empty script

First we need to create an empty script, and we will place that in a personal folder inside the SCRIPTS folder:

- Create a folder with your name in the SCRIPTS folder. I created a folder named

MY_SCRIPTS.

- In the folder SCRIPTS/MY_SCRIPTS I create a new file “my_script.m” . Only use characters a-Z, A-Z and “_”. Spaces and other strange characters are not allowed, but use “_” instead of a space (you will see why later on).
- Double-click on my_script.m to open it in Matlab.

Note: If you create a new script, it will not be recognized from the already open imlook4d windows. You need to create a new imlook4d window, either by opening a new image file, or by selecting the imlook4d menu “/SCRIPTS/Change/Duplicate”.

Basics – access the data

The aim now is to get a feel for how scripting works. First open an image file (as described in a previous tutorial). Select the SCRIPTS menu, and let go.

Select the main Matlab window and look in the variable list. You will now see two variables exported to the Matlab workspace:

```
imlook4d_current_handle (handle to the current imlook4d instance)
imlook4d_current_handles      (handles structure to GUI and data)
```

These two variables will be used by the scripts language that will be introduced below.

Now, edit the my_script.m file you created earlier. Write the following, and save:

```
StartScript;
```

Select SCRIPTS/MY_SCRIPTS from the imlook4d window, and you will see a duplicate of the imlook4d window open. You will also see many more variables in the Matlab workspace:

```
handleToOriginal, imlook4d_Cdata, imlook4d_ROI, imlook4d_ROINames,
imlook4d_ROI_number, imlook4d_current_handle, imlook4d_current_handles,
imlook4d_duration, imlook4d_frame, imlook4d_slice, imlook4d_time,
imlook4d_variables_before_script
```

These variables gives direct access to most things you need to do to manipulate image volumes, and region-of-interests (ROIs). The duplicate imlook4d menu is where your manipulated data will be written back.

Basics – make your first calculation

Lets now make a script that multiplies all pixels by 2. Edit the script, and type:

```
StartScript
imlook4d_Cdata=2*imlook4d_Cdata;
WindowTitle('My_Script did this !');
EndScript
```

You should now get a duplicate imlook4d window, but with the color scale showing twice the maximum value. Also the imlook4d window title is changed to “My_Script did this!”.

Basics – meaning of variables

The table below shows what variables are Exported when creating a script. This is exactly the same as selecting the imlook4d menu “/Workspace/Export untouched”. The table also shows what variables are imported back with the imlook4d menu “/Workspace/Import”. All these variables are referring to the new imlook4d window that was opened.

Variable	Export	Import	Explanation
imlook4d_current_handle	x		Handle to current imlook4d
imlook4d_current_handles	x	x	Struct defining everything in current imlook4d
imlook4d_time	x	x	Frame time
imlook4d_duration	x	x	Frame duration
imlook4d_Cdata	x	x	4D pixel data
imlook4d_ROINames	x	x	ROI names
imlook4d_ROI	x	x	ROI pixels in 3D matrix
imlook4d_slice	x		Current slice
imlook4d_frame	x		Current frame
imlook4d_ROI_number	x		Current selected ROI
imlook4d_store			Mechanism for storing values in session. See descriptions

Other variables are more for administrative tasks, such as cleaning up after a script is finished. The cleaning up is performed as part of the `EndScript` command.

Understanding your first script

When you did your first script, you noted that a new imlook4d window opened, and that a number of variables were written. I will now explain what happened, and the meaning of the variables. The best way to understand this is to rewrite the script a bit, so that "`my_script.m`" looks like this:

```
StartScript
imlook4d_Cdata=2*imlook4d_Cdata;
WindowTitle('My_Script did this !');
%EndScript
Import % This imports data,but doesn't remove variables from workspace
```

We now have two imlook4d windows. The window and all the data is called an instance. Each imlook4d window has a handle (like a variabel pointing to it)

```
HandleToOriginal            points to the original imlook4d window
imlook4d_current_handle     points to the new imlook4d window
```

We also have a structure `imlook4d_current_handles` containing all data from the imlook4d window. This you normally don't need to touch, but it gives you access to the GUI, and to the data. The data which is shown in the workspace should be used instead of accessing it through the structure, since these variables takes precedence to the structure when importing.

There is also a variable `imlook4d_variables_before_script` that keeps track of which variables existed before the script was initiated, so that the `EndScript` command knows what variables should be cleaned at the end of the script.

Example script skeletons

The following examples show a few cases you can start scripting from.

Example 1. Create one new imlook4d window, and operate on

```
% Start script
```

```
    StartScript;

    % process with your own code
    % add new data to variables
    % possibly modify

    % Finish script
    EndScript
```

Example 2. Create multiple new imlook4d windows from the orginal, and modify the data in each

```
 % Start script
    StartScript;

    % Create first imlook4d
    imlook4d_Cdata=2*imlook4d_Cdata;      %Calculate
    WindowTitle('(2xdata) ','prepend')    %Add "(2xdata) " to original title
    Import

    % Create second imlook4d
    DuplicateOriginal;
    imlook4d_Cdata=3*imlook4d_Cdata;
    WindowTitle('(3xdata) ','prepend')    %Add "(3xdata) " to original title
    Import

    % Create last imlook4d
    DuplicateOriginal;
    imlook4d_Cdata=4*imlook4d_Cdata;
    WindowTitle('(4xdata) ','prepend') %Add "(4xdata) " to original title
    Import %This is not necessary - EndScript does an import as well

    % Finish script
    EndScript                           %Clean up
```

Example 3. A typical script (doing the same as the first script), with more options for modifications, would looks as follows. For instance, you can turn on/off parts of what is normally done in StartScript and EndScript:

```
% StartScript
    % Store variables (so we can clear all variables created in this script)
    StoreVariables;

    % Make a duplicate to work on
    Duplicate       % Make a copy of imlook4d instance in handle newHandle
    MakeCurrent     % Rename newHandle to imlook4d_current_handle
    Export          % Export variables from current imlook4d instance

% Your code goes here
    WindowTitle('SUV','prepend')

    % process with your own code
    % add new data to variables
    % possibly modify
     ...

% EndScript
    % Import data (variables, and imlook4d_current_handles)
    Import

    % Clean up  variables created in this script
    ClearVariables
```

List of Script commands

The following simple scripting commands are available

StartScript	Starts a script with this command exports the variables in the table below, and creates a new imlook4d window to work in. The `imlook4d_current_handle` and `imlook4d_current_handles` are set to point to the this new window. Hotelling filtering is applied to the imlook4d_Cdata image matrix. Interpolations, thresholding and window levels are **not** modifying the exported variables. `handleToOriginal` is set to `imlook4d_current_handle` , so that initiating imlook4d-window can be referenced later in a script. (Startscript uses the below commands internally: `StoreVariables, Duplicate, MakeCurrent, Export)`
EndScript	Imports the modified variables, and modified `imlook4d_current_handles` into the window defined in `StartScript`. The window title is appended with the string in the variable `historyDescriptor`. It also cleans up the variables created in the script (from StartScript onwards). (EndScript uses the below commands: `Title, Import, ClearVariables)`

WindowTitle	`WindowTitle(str)`
	sets the title of the current window to the string `str`.
	
	`WindowTitle(str,'prepend')`
	sets the string `str` before the title of the current window.
	
	`WindowTitle(str,'append')`
	sets the string `str` after the title of the current window.

DuplicateOriginal	This duplicates the imlook4d window that started the script. It also exports the variables (just as the Export command would). This is useful if multiple windows will be created.
TACT	Generates time-activity data from the ROIs. These are stored in the workspace variables: tact.activity (columns with mean activity in each ROI) tact.n (number of pixels in each ROI) tact.stdev (columns with standard deviation within each ROI)
LoadROI	`LoadROI(str)` Loads a ROI from file path `str`
MakeROI	`MakeROI(str)`

Creates a ROI with the name `str`.

```
a=MakeROI(str)
```

Creates a ROI with the name `str`, and returns the ROI-number for the created ROI.

SelectROI — `SelectROI(number)` or `SelectROI(name)`
Selects ROI with ROI-number or ROI-name.

ExportROIs — Export ROIs to workspace variable imlook4d_ROI_data.
Examples showing how to use this data:

```
imlook4d_ROI_data.midtime(1)        % Gets midtime of frame 1
imlook4d_ROI_data.duration(1)       % Get duration of frame 1
imlook4d_ROI_data.Npixels(1)        % Get number of pixels in ROI 1
imlook4d_ROI_data.mean(3,1)         % Mean activity in frame 3, ROI 1
imlook4d_ROI_data.mean(:,1)         % Mean activity in each frame of dyn scan, ROI 1
imlook4d_ROI_data.stdev(3,1)        % St. dev. of pixels in frame 3, ROI 1
imlook4d_ROI_data.pixels{1}(:,5)    % All pixel values in frame 5, ROI 1
imlook4d_ROI_data.max(3,1)          % Max pixel value in frame 3, ROI 1
imlook4d_ROI_data.min(3,1)          % Min pixel value in frame 3, ROI 1
imlook4d_ROI_data.volume(1)         % Volume of ROI 1
imlook4d_ROI_data.voxelsize         % Voxelsizes [x y z dV] (dV=voxel volume=x*y*z)
imlook4d_ROI_data.centroid{1}.x     % Centroid x position, of ROI 1
imlook4d_ROI_data.dimension{1}.y    % Highest width in y-directin of ROI 1
```

SelectWindow — Pops up a dialog with a text message, to select another (imlook4d) window.
Click OK when you have activated the imlook4d-window you want to use. Returns a handle to the selected window.
Useage:

```
handle=SelectWindow(message)
```

Example (multi-line message when entered as cells):

```
handle=SelectWindow({...
'Select template image (from imlook4d/Windows menu)', ...
'(image that we want slices to match' ...
});
```

StoreValues — `StoreValues( name, valuesToStore)`
Mechanism to storing a cell array of string values during this session. The value set is identified by `name`. The stored values are deleted when calling `clear all`, or clear `imlook4d_store`

RetrieveEarlierValues — `outCellArray = RetriveEarlierValues( name, defaultValues)`
Returns a cell array of strings, that was stored under the identifier name. If no values are stored, the defaultValues cell array is returned, and also stored as the one returned next time calling `RetrieveEarlierValues`.

(Note: all script names start with a capital letter)

Specialized script commands

The following commands are performing smaller functionality, but may also be useful (they are actually used within the above commands):

Open — `Open, Open(arg) or handle=Open(arg).` Opens an imlook4d instance, and creates variables (`imlook4d_current_handle,`

	`imlook4d_current_handles`) in workspace. Same as menu File/Open followed by clicking the SCRIPTS menu. The input argument arg, could be a matrix, or a complete file path. The optional output argument is a handle to the imlook4d instance.
Save	Save dialog on current data (what is in `imlook4d_current_handles`)
Export	exports image matrix as viewed on screen. Also exports useful variables from `imlook4d_current_handle` (same as menu "Workspace/export filtered". Also the handles structure `imlook4d_current_handles` is exported.
ExportUntouched	exports image matrix (no interactive filters are applied). Also exports useful variables from `imlook4d_current_handle` (same as menu "Workspace/export untouched"). Also the handles structure `imlook4d_current_handles` is exported.
Import	imports variables (same as menu "Workspace/import") into `imlook4d_current_handle`. Also the handles structure `imlook4d_current_handles` is imported. Variables such as `imlook4d_Cdata` takes precedence over the same variable in the handles structure (e.g. `imlook4d_current_handles.image.Cdata`).
ImportUntouched	imports variables (same as menu "Workspace/import"), except that the matrix `imlook4d_Cdata` is not imported
Title	adds the content you put in variable `historyDescriptor` to the window title of `imlook4d_current_handle`. This is going to be rolled out, and replaced by the WindowTitle function above.
Duplicate	duplicates the current window, and creates `newHandle, newHandles`. These have the same function as `imlook4d_current_handle`, and `imlook4d_current_handles`, but for the new window.
MakeCurrent	makes the `newHandle, newHandles` the current handles (renaming them to `imlook4d_current_handle, imlook4d_current_handles`). Following this command, scripts will operate on these. `newHandle, newHandles` are deleted.
StoreVariables	Remember variables that exist in workspace before script is executed. The variable names are stored in `imlook4d_variables_before_script`. A possible problem may occur in the following condition:

Script A calls StoreVariables.
Script A calls script B
Script B calls StoreVariables
Script B calls ClearVariables – Here Script A will loose track of stored variables

The remedy for this is to keep track of the stored variables yourself in script A:

```
temp_list=imlook4d_variables_before_script;  % Make temporal list
Call script B
imlook4d_variables_before_script=temp_list;  % Restore list
```

ClearVariables	Clear variables that were created in script after that `StoreVariables` was called