A SOFTWARE BASED APPROACH TO DEMULTIPLEX IMAGE INTO ITS CONSTITUENT WAVELENGTH COMPONENTS

By Chetan Rao

A report submitted in partial fulfilment of the requirements for Research credits in

Electrical/Telecommunications Engineering

Drexel University

Date: 9th June 2011
A report presented on a software based approach is put forward for de-multiplexing a composite wavelength image into its various wavelength components for Hyperspectral Imaging. The existing architecture to capture Hyperspectral data uses diffraction gratings is a hardware approach.

The main aim of the study is to put forward a low cost, fast and flexible software based approach to de-multiplex image based on the presence of the constituent wavelength components.
# Table of Contents

**Chapter I**: Introduction to Hyperspectral Imaging and its Applications ............................. 1  
**Chapter II**: Case Study ........................................................................................................ 5  
**Chapter III**: Methodology and Results ............................................................................... 6  
**Chapter IV**: GUI: Hyperspec Demux Toolbox .................................................................... 9  
**Chapter V**: MATLAB SCRIPTS ....................................................................................... 10  
**Bibliography** ....................................................................................................................... 23
ACKNOWLEDGMENTS

The author wishes to express sincere appreciation to Dr. Adam for their assistance in the preparation of this manuscript. In addition, special thanks to Dr. Sameet Shriyan whose familiarity with the needs and ideas were helpful during the early programming phase of this undertaking.
Chapter 1

INTRODUCTION TO HYPERSPECTRAL IMAGING

AND

ITS APPLICATIONS

Hyperspectral imaging collects and processes information from across the electromagnetic spectrum. Much as the human eye sees visible light in three bands (red, green, and blue), spectral imaging divides the spectrum into many more bands. This technique of dividing images into bands can be extended beyond the visible.

Hyperspectral images contain a wealth of data, but interpreting them requires an understanding of exactly what properties of ground materials we are trying to measure, and how they relate to the measurements actually made by the Hyperspectral sensor. Images acquired simultaneously in many narrow, adjacent wavelength bands. Set of brightness values for a single raster cell position in the Hyperspectral image.
Hyperspectral sensors collect information as a set of 'images'. Each image represents a range of the electromagnetic spectrum and is also known as a spectral band. These 'images' are then combined and form a three-dimensional hyperspectral data cube for processing and analysis.

Hyperspectral cubes are generated from airborne sensors like the NASA's Airborne Visible/Infrared Imaging Spectrometer (AVIRIS), or from satellites like NASA's Hyperion. However, for many development and validation studies, handheld sensors are used.

The precision of these sensors is typically measured in spectral resolution, which is the width of each band of the spectrum that is captured. If the scanner detects a large number of fairly narrow frequency bands, it is possible to identify objects even if they are only captured in a handful of pixels. However, spatial resolution is a factor in addition to spectral resolution. If the pixels are too large, then multiple objects are captured in the same pixel and become difficult to identify. If the pixels are too small, then the energy captured by each sensor cell is low, and the decreased signal-to-noise ratio reduces the reliability of measured features.
APPLICATIONS OF HYPERSPECTRAL IMAGING

Hyperspectral remote sensing is used in a wide array of real-life applications. Although originally developed for mining and geology (the ability of Hyperspectral imaging to identify various minerals makes it ideal for the mining and oil industries, where it can be used to look for ore and oil) it has now spread into fields as widespread as ecology and surveillance. This technology is continually becoming more available to the public, and has been used in a wide variety of ways. Organizations such as NASA have catalogues of various minerals and their spectral signatures, and have posted them online to make them readily available for researchers.

Agriculture

Although the costs of acquiring hyperspectral images is typically high, for specific crops and in specific climates hyperspectral remote sensing is used more and more for monitoring the development and health of crops. In Australia work is underway to use imaging spectrometers to detect grape variety, and develop an early warning system for disease outbreaks. Furthermore work is underway to use hyperspectral data to detect the chemical composition of plants which can be used to detect the nutrient and water status of wheat in irrigated systems.

Mineralogy

Hyperspectral remote sensing of minerals is well developed. Many minerals can be identified from airborne images, and their relation to the presence of valuable minerals such as gold and diamonds is well understood. Currently the move is towards understanding the relation between oil and gas leakages from pipelines and natural wells; their effect on the vegetation and the spectral signatures.

A set of stones scanned with a Specim LWIR-C imager in the thermal infrared range from 7.7 μm to 12.4 μm. The quartz and feldspar spectra are clearly recognizable.
Hyperspectral thermal infrared emission measurement, an outdoor scan in winter conditions, ambient temperature -15°C. Relative radiance spectra from various targets in the image are shown with arrows. The infrared spectra of the different objects such as the watch class have clearly distinctive characteristics. The contrast level indicates the temperature of the object. Image produced with a Specim LWIR hyperspectral imager.

Hyperspectral surveillance is the implementation of hyperspectral scanning technology for surveillance purposes.
Chapter II

CASE STUDY

Statement of Problem

A report presented on a software based approach is put forward for de-multiplexing a composite wavelength image into its various wavelength components for Hyperspectral Imaging. The existing architecture to capture Hyperspectral data uses diffraction gratings is a hardware approach.

Mathematically:

$$F(\lambda_1, \lambda_2, \lambda_3) = F(\lambda_1) + F(\lambda_2) + F(\lambda_3)$$

Where,

$F(\lambda_1, \lambda_2, \lambda_3)$ is a Multiplexed image: Single image having set of the wavelengths: 400 - 800 nm

$F(\lambda_1), ...$ are the De-multiplexed images which correspond to constituent wavelengths

Purpose of Study

The main purpose of the study is to put forward a low cost, fast and flexible software based approach to de-multiplex image based on the presence of the constituent wavelength components which can be used in Hyperspectral imaging.
Chapter III

METHODOLOGY

Hyperspectral image data is captured for 10 seconds at a sampling frequency corresponding to the capture rate of camera. Fourier transform (N-FFT) of the data is calculate to analyse the spectra of the multiplexed signal obtained from a pixel. The captured data is such that the HPDLC’s perform the filtering of the wavelengths as per the switching frequency of the filter. Each filter switching frequency $f_i$ is mapped to a specific wavelength. Using peaks in the spectra, the presence of filter switching frequency is detected. Presence of the filter switching frequency implies the presence of the corresponding wavelength. Using the wavelength and switching frequency mapping the pixel values for the wavelength components can be retrieved.

Example of noisy multiplexed signal (20 components) obtained at pixel:
Spectra of the multiplexed signal with peaks shown using red vertical lines.

Channel Vs Intensity Graph: Each channel corresponding to a wavelength.

Channel # = Wavelength Intensity in increasing order of Filter Frequencies.
Observe the absence of Channel 4 which gives an accuracy of identification of wavelengths in the multiplexed image = 95%.

Accuracy analysis with variation in number of filters (No_Signal). Time required at pixel level to identify the wavelengths present in the multiplexed image.

<table>
<thead>
<tr>
<th>No_Signal</th>
<th>Accuracy</th>
<th>Time</th>
<th>No_Signal</th>
<th>Accuracy</th>
<th>Time</th>
<th>No_Signal</th>
<th>Accuracy</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>93.5</td>
<td>0.160386</td>
<td>20</td>
<td>95</td>
<td>0.061848</td>
<td>1000</td>
<td>99.3</td>
<td>0.545935</td>
</tr>
<tr>
<td>200</td>
<td>95</td>
<td>0.111484</td>
<td>20</td>
<td>95</td>
<td>0.023272</td>
<td>1000</td>
<td>99.2</td>
<td>0.584473</td>
</tr>
<tr>
<td>200</td>
<td>94</td>
<td>0.111973</td>
<td>20</td>
<td>90</td>
<td>0.022006</td>
<td>1000</td>
<td>99.8</td>
<td>0.576649</td>
</tr>
<tr>
<td>200</td>
<td>92</td>
<td>0.111871</td>
<td>20</td>
<td>80</td>
<td>0.022892</td>
<td>1000</td>
<td>99.5</td>
<td>0.53113</td>
</tr>
<tr>
<td>200</td>
<td>93</td>
<td>0.117623</td>
<td>20</td>
<td>95</td>
<td>0.024164</td>
<td>1000</td>
<td>96.8</td>
<td>0.630426</td>
</tr>
<tr>
<td>200</td>
<td>89.5</td>
<td>0.112534</td>
<td>20</td>
<td>95</td>
<td>0.022507</td>
<td>1000</td>
<td>98.1</td>
<td>0.495999</td>
</tr>
<tr>
<td>200</td>
<td>96.5</td>
<td>0.112983</td>
<td>20</td>
<td>70</td>
<td>0.02245</td>
<td>1000</td>
<td>99</td>
<td>0.510343</td>
</tr>
<tr>
<td>200</td>
<td>93.5</td>
<td>0.124024</td>
<td>20</td>
<td>95</td>
<td>0.022242</td>
<td>1000</td>
<td>97.9</td>
<td>0.484224</td>
</tr>
<tr>
<td>200</td>
<td>90</td>
<td>0.116042</td>
<td>20</td>
<td>90</td>
<td>0.022185</td>
<td>1000</td>
<td>100</td>
<td>0.496286</td>
</tr>
<tr>
<td>200</td>
<td>90.5</td>
<td>0.137237</td>
<td>20</td>
<td>100</td>
<td>0.022333</td>
<td>1000</td>
<td>98.1</td>
<td>0.501521</td>
</tr>
<tr>
<td>200</td>
<td>92.5</td>
<td>0.116813</td>
<td>20</td>
<td>90</td>
<td>0.022534</td>
<td>1000</td>
<td>99.1</td>
<td>0.498176</td>
</tr>
<tr>
<td>200</td>
<td>91</td>
<td>0.127986</td>
<td>20</td>
<td>100</td>
<td>0.030798</td>
<td>1000</td>
<td>96.9</td>
<td>0.491832</td>
</tr>
<tr>
<td>200</td>
<td>90.5</td>
<td>0.112911</td>
<td>20</td>
<td>90</td>
<td>0.024634</td>
<td>1000</td>
<td>98.4</td>
<td>0.532928</td>
</tr>
<tr>
<td>200</td>
<td>94</td>
<td>0.117469</td>
<td>20</td>
<td>90</td>
<td>0.022882</td>
<td>1000</td>
<td>100</td>
<td>0.530726</td>
</tr>
<tr>
<td>200</td>
<td>89</td>
<td>0.125949</td>
<td>20</td>
<td>85</td>
<td>0.024033</td>
<td>1000</td>
<td>98</td>
<td>0.525301</td>
</tr>
<tr>
<td>200</td>
<td>94</td>
<td>0.116045</td>
<td>20</td>
<td>100</td>
<td>0.022391</td>
<td>1000</td>
<td>99.2</td>
<td>0.496802</td>
</tr>
<tr>
<td>200</td>
<td>90.5</td>
<td>0.113271</td>
<td>20</td>
<td>95</td>
<td>0.025013</td>
<td>1000</td>
<td>97.9</td>
<td>0.492365</td>
</tr>
<tr>
<td>200</td>
<td>95</td>
<td>0.129381</td>
<td>20</td>
<td>90</td>
<td>0.02255</td>
<td>1000</td>
<td>99.1</td>
<td>0.496382</td>
</tr>
<tr>
<td>200</td>
<td>91</td>
<td>0.120483</td>
<td>20</td>
<td>100</td>
<td>0.024831</td>
<td>1000</td>
<td>97.3</td>
<td>0.505904</td>
</tr>
<tr>
<td>200</td>
<td>94</td>
<td>0.113055</td>
<td>20</td>
<td>85</td>
<td>0.026635</td>
<td>1000</td>
<td>97.3</td>
<td>0.50011</td>
</tr>
</tbody>
</table>

Further Research

- Image De-multiplexing code optimization.
- Implement multi-frequency identification and amplitude extraction technique.
- Test with real data.
- Improve accuracy of identification of the wavelengths in a multiplexed image.
- Improve on the features of the Hyperspec Demux Toolbox GUI.
Chapter IV

GUI: HYPERSPEC DEMUX TOOLBOX

Hyperspec Demux Toolbox GUI to input the Hyperspectral data

Image Display by slider GUI to view the De-multiplexed image
Vertical scroll bar gives access to view the various WAVELENGTH components
Horizontal scroll bar gives access to view the wavelength components at different TIME instants.
Chapter V

MATLAB SCRIPTS

Hyperspec Demux Toolbox

function varargout = DemuxGUIwoslide(varargin)

% DEMUXGUIWOSLIDE M-file for DemuxGUIwoslide.fig
% DEMUXGUIWOSLIDE, by itself, creates a new DEMUXGUIWOSLIDE or raises the
% existing
% singleton.*.
% H = DEMUXGUIWOSLIDE returns the handle to a new DEMUXGUIWOSLIDE or the
% handle to
% the existing singleton.*.
% DEMUXGUIWOSLIDE('CALLBACK',hObject,eventData,handles,...) calls the local
% function named CALLBACK in DEMUXGUIWOSLIDE.M with the given input
% arguments.
% DEMUXGUIWOSLIDE('Property','Value',...) creates a new DEMUXGUIWOSLIDE or
% raises the
% existing singleton*. Starting from the left, property value pairs are
% applied to the GUI before DemuxGUIwoslide_OpeningFcn gets called. An
% unrecognized property name or invalid value makes property application
% stop. All inputs are passed to DemuxGUIwoslide_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

% Begin initialization code - DO NOT EDIT

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                    'gui_Singleton',  gui_Singleton, ...
                    'gui_OpeningFcn', @DemuxGUIwoslide_OpeningFcn, ...
                    'gui_OutputFcn',  @DemuxGUIwoslide_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

% --- Executes just before DemuxGUIwoslide is made visible.
function DemuxGUIwoslide_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    reserved - to be defined in a future version of MATLAB
% varargin   reserved - to be defined in a future version of MATLAB

% --- Scripts the functions that make the GUI respond to buttons
function myguiCallback(hObject, eventdata, handles)
% handles structure with handles and user data (see GUIDATA)
% varargin command line arguments to DemuxGUIwoslide (see VARARGIN)

% Choose default command line output for DemuxGUIwoslide
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes DemuxGUIwoslide wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = DemuxGUIwoslide_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function edit1_Callback(hObject, eventdata, handles)
% hObject handle to edit1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
% str2double(get(hObject,'String')) returns contents of edit1 as a double
global row
row = str2num(get(hObject,'String'));
% checks to see if input is empty. if so, default edit1_Callback to zero
if (isempty(row))
    set(hObject,'String','4')
end
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject handle to edit1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc & isequal(get(hObject,'BackgroundColor'),
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit2_Callback(hObject, eventdata, handles)
% hObject handle to edit2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
% str2double(get(hObject,'String')) returns contents of edit2 as a double
global Fs
Fs = str2num(get(hObject, 'String'));

% checks to see if input is empty. if so, default edit2_Callback to zero
if (isempty(Fs))
    set(hObject, 'String', '30')
end
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'),
    get(0, 'defaultUicontrolBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double
global Time
Time = str2num(get(hObject, 'String'));

% checks to see if input is empty. if so, default edit3_Callback to zero
if (isempty(Time))
    set(hObject, 'String', '10')
end
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'),
    get(0, 'defaultUicontrolBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double
global Filter_Freq
Filter_Freq = str2num(get(hObject, 'String'));
% checks to see if input is empty. if so, default edit4_Callback to zero
if (isempty(Filter_Freq))
    set(hObject,'String','2,7,11,13,15')
end
guida(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double
global column
column = str2num(get(hObject,'String'));

% checks to see if input is empty. if so, default edit1_Callback to zero
if (isempty(column))
    set(hObject,'String','4')
end
guida(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a double
global Wavelengths
Wavelengths = str2num(get(hObject,'String'));

% checks to see if input is empty. if so, default edit6_Callback to zero
if (isempty(Wavelengths))
    set(hObject,'String','400,500,600,710,740')
end
function edit6_CreateFcn(hObject, eventdata, handles)
    % hObject    handle to edit6 (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    empty - handles not created until after all CreateFcns called

    % Hint: edit controls usually have a white background on Windows.
    % See ISPC and COMPUTER.
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
end

function edit7_Callback(hObject, eventdata, handles)
    % hObject    handle to edit7 (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)

    % Hints: get(hObject,'String') returns contents of edit7 as text
    %        str2double(get(hObject,'String')) returns contents of edit7 as a double
    global No_Filter
    No_Filter = str2num(get(hObject,'String'));

    % checks to see if input is empty. if so, default edit7_Callback to zero
    if (isempty(No_Filter))
        set(hObject,'String','5')
    end
end

function edit8_Callback(hObject, eventdata, handles)
    % hObject    handle to edit8 (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)

    % Hints: get(hObject,'String') returns contents of edit8 as text
    %        str2double(get(hObject,'String')) returns contents of edit8 as a double
    global Wave_intensity
    Wave_intensity = str2num(get(hObject,'String'));

    % checks to see if input is empty. if so, default edit5_Callback to zero
    if (isempty(Wave_intensity))
        set(hObject,'String','50, 76, 80, 111, 210')
    end
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties. 
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of radiobutton1
set(handles.radiobutton1,'value',1);

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all;

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc;
global row column Fs No_Filter Time Filter_Freq Wavelengths Wave_intensity
if get(handles.radiobutton1,'value') == 1
    demux(row,column,Fs,No_Filter,Time,Filter_Freq,Wavelengths,Wave_intensity);
    imagedisplaybyslider
else
    display('Select one radio options');
end

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.edit1,'String','4');
set(handles.edit2,'String','30');
set(handles.edit3,'String','10');
set(handles.edit4,'String','2 7 11 13 15');
set(handles.edit5,'String','4');
set(handles.edit6,'String','400 500 600 710 740');
set(handles.edit7,'String','5');
set(handles.edit8,'String','50 76 80 111 210');
**Imagedisplay by slider**

```matlab
function varargout = imagedisplaybyslider(varargin)
% IMAGEDISPLAYBYSLIDER M-file for imagedisplaybyslider.fig
% IMAGEDISPLAYBYSLIDER, by itself, creates a new IMAGEDISPLAYBYSLIDER or
% raises the existing
% singleton*.
% H = IMAGEDISPLAYBYSLIDER returns the handle to a new IMAGEDISPLAYBYSLIDER
% or the handle to
% the existing singleton*.
% IMAGEDISPLAYBYSLIDER('CALLBACK',hObject,eventData,handles,...) calls the
% local
% function named CALLBACK in IMAGEDISPLAYBYSLIDER.M with the given input
% arguments.
% IMAGEDISPLAYBYSLIDER('Property','Value',...) creates a new
% IMAGEDISPLAYBYSLIDER or raises the
% existing singleton*. Starting from the left, property value pairs are
% applied to the GUI before imagedisplaybyslider_OpeningFunction gets
% called. An
% unrecognized property name or invalid value makes property application
% stop. All inputs are passed to imagedisplaybyslider_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

% Copyright 2002-2003 The MathWorks, Inc.

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

% Last Modified by GUIDE v2.5 26-Mar-2008 00:30:28

% Begin initialization code - DO NOT EDIT

    gui_Singleton = 1;
    gui_State = struct('gui_Name', mfilename, ...
                       'gui_Singleton', gui_Singleton, ...
                       'gui_OpeningFcn', @imagedisplaybyslider_OpeningFcn, ...
                       'gui_OutputFcn', @imagedisplaybyslider_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

% --- Executes just before imagedisplaybyslider is made visible.
function imagedisplaybyslider_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to imagedisplaybyslider (see VARARGIN)

% Choose default command line output for imagedisplaybyslider
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes imagedisplaybyslider wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = imagedisplaybyslider_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% ............the following code edited by user
% the vertical slider
global Time
global Fs
global No_Filter

set(handles.slider2,'value','1');
set(handles.slider2,'max','Fs*Time');
set(handles.slider2,'min','1');
set(handles.slider2,'SliderStep',[1/(Fs*Time-1) 10/(Fs*Time-1)]);
% the horizontal slider
set(handles.slider1,'value','1');
set(handles.slider1,'max','No_Filter');
set(handles.slider1,'min','1');
set(handles.pushbutton1,'string','Reset');
set(handles.slider1,'SliderStep',[1/(No_Filter-1) 1/(No_Filter-1)]);

plotfunc

%.................................................................
% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
% get(hObject,'Min') and get(hObject,'Max') to determine range of slider
% ............the following code edited by user
plotfunc
%.................................................................
% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)

% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background, change 'usewhitebg' to 0 to use default. See ISPC and COMPUTER.
usewhitebg = 1;
if usewhitebg
    set(hObject, 'BackgroundColor', [.9 .9 .9]);
else
    set(hObject, 'BackgroundColor', get(0, 'defaultUicontrolBackgroundColor'));
end

% ..........the following code edited by user
global Time
global Fs
global No_Filter
set(handles.slider1, 'value', 1);
set(handles.slider1, 'max', No_Filter);
set(handles.slider1, 'min', 1);
set(handles.slider1, 'SliderStep', [1/(No_Filter - 1) 1/(No_Filter - 1)]);

% ............................................

% --- Executes on slider movement.
function slider2_Callback(hObject, eventdata, handles)

% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider

% ..........the following code edited by user
plotfunc

% ............................................

% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)

% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background, change 'usewhitebg' to 0 to use default. See ISPC and COMPUTER.
usewhitebg = 1;
if usewhitebg ==1
    set(hObject, 'BackgroundColor', [.9 .9 .9]);
else
    set(hObject, 'BackgroundColor', get(0, 'defaultUicontrolBackgroundColor'));
end

% ..........the following code edited by user
global Time
global Fs
global No_Filter
set(handles.slider2, 'value', 1);
set(handles.slider2, 'max', Fs*Time);
set(handles.slider2, 'min', 1);
set(handles.slider2, 'SliderStep', [1/(Fs*Time - 1) 1/(Fs*Time - 1)]);
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% ........the following code edited by user
set(handles.slider1,'value',1);
set(handles.slider1,'max',No_Filter);
set(handles.slider1,'min',1);
set(handles.slider1,'SliderStep',[1/(No_Filter-1) 1/No_Filter(No_Filter-1)]);
set(handles.slider2,'value',1);
set(handles.slider2,'max',Fs*Time);
set(handles.slider2,'min',1);
set(handles.slider2,'SliderStep',[1/(Fs*Time-1) 1/(Fs*Time-1)]);

plotfunc
%..............................................
Demux code

```matlab
function demux(r,c,Fs,No_Filter,Time,Filter_Freq,Wavelengths,Wave_intensity)
% Signal Processing on the Captured image frames.
clc;
%z=1;
global time
for row=1:r
    for column=1:c
        t=0:1/Fs:Time;'% No. of samples obtained
        noise=randn(1,length(t));
        % Signals generated at frequencies "Filter_Freq" corresponding
        % to
        % WAVELENGTHS
        No_signals = No_Filter;
        amp=zeros(1,No_signals);
        amp=Wave_intensity;
        n=length(t); % number of Samples
        Signal=amp' * ones(1,n);
        NFilter=No_Filter;
        char s;
        Sin=sort(Filter_Freq,'ascend');
        s=num2str(Sin);

        % Filter signals in the form of square waves at "Filter_Freq"
        Filter=(1+square(2*pi.*repmat(Filter_Freq,length(t),1).*repmat(t,length(Filter_Freq),1)))/2 + ones(No_signals,1)*noise;
        %figure(2); subplot(5,1,1); plot(Filter(1,:)); subplot(5,1,2); plot(Filter(2,:)); subplot(5,1,3); plot(Filter(3,:)); subplot(5,1,4); plot(Filter(4,:)); subplot(5,1,5); plot(Filter(5,:)); title('Filter Signals');
        % Filtered Multiplexed Signal
        Filt_Mux_Signal=ceil(mean((Filter.*Signal),1));
        % figure(3); plot(t,Filt_Mux_Signal);title('Filtered Multiplexed Signal');
        xlabel('Time')
        ylabel('Intensity Level')
        T = 1/Fs; % Sample time
        L = n; % Length of signal
        t = (0:L-1)*T; % Time vector
        % Sum of a 50 Hz sinusoid and a 120 Hz sinusoid
        NFFT = 2^(nextpow2(L)); % Next power of 2 from length of y
        % psd(Filt_Mux_Signal);
        Y = fft(Filt_Mux_Signal,NFFT)/L;
        f = Fs/2*linspace(0,1,NFFT/2+1);
        [v i]=find(f>1);
        ind=min(i);
        spectrum=2*abs(Y(1:NFFT/2+1));
        % Plot single-sided amplitude spectrum.
        % figure(4); plot(f(ind:length(f)),spectrum(ind:length(f))); %set(gca, 'XTick',1:b);
        % Intensity Vs Channel plot
        % title('Spectrum filter signals are square waves: ');
        % xlabel('Frequency (Hz)');
        % ylabel('|Y(f)|');
    end
end
```
indice=findpeaks(spectrum);
peakf=f(indice);

% hold on; plot(peakf,spectrum(indice),'x');
  for i=1:length(Sin)
    plot([Sin(i) Sin(i)],[0 max(spectrum(ind:liend(f)))]','r');
  end
% Hold off;
  i=find((peakf>1));
  channels=truncdec(peakf(i),0);
  [filter_freq indamp] = sort(truncdec(Filter_freq,1),'ascend');
  amp=amp(indamp);
  classification=ismember(filter_freq,channels);
  channelpresent=find(classification==1);
  %figure(5);
  if isempty(channelpresent)
    bar(amp); title('None of the Wavelengths are present');
  else
    bar(amp .* classification); title('Channel # = Wavelength Intensity in increasing order of Filter Frequencies');
    set(gca, 'XTick',1:2:No_signals);
    xlabel('Channel #'); ylabel('Intensity');
  end
accuracy=100*count(classification,'==1')/length(filter_freq);
  %display(accuracy);
  %display(classification);
  %table(z,:)=[No_signals accuracy t];
  % Constituent images
  for ins=1:length(t)
    for x=1:No_signals
      if classification(x)==1
        time(ins).img(row,column,x)= Signal(x,ins);
      else
        time(ins).img(row,column,x)= 0;
      end
    end
    %z=z+1;
  end
assignin('base','time');
%table
%xlsxwrite('table.xlsx',table);
function plotfunc

    global Wavelengths Fs time

% the vertical slider
rr=findobj(gcf,'Tag','slider1');
r1=get(rr,'value');

% the horizontal slider
rr=findobj(gcf,'Tag','slider2');
r2=get(rr,'value');
xlabel(['Wavelength: ' num2str(Wavelengths(ceil(r1))) ' and Time Instant: ' num2str(r2/Fs)]);
temp_matrix=time(ceil(r2)).img(:,:,ceil(r1));

image(temp_matrix);
colormap('gray');
axis off;
BIBLIOGRAPHY
