PAN FUSION OF A MULTISPECTRAL IMAGE

Documentation

SIP Project Topic (2017 – 2018)

Submitted By:-
Aarif Shaikh(173310007)
Ravi Pratap Singh (173314001)
Harshula Tulapurkar (174318001)

Table of Contents

Introduction

In Satellite imagery, there is a tradeoff involved when it comes to the spatial and spectral information captured by the sensors. If we break down a wavelength band into discrete regions and capture them, we end up with a low spatial resolution, whereas if we incorporate information from multiple such contiguous intervals, we end up with a higher spatial resolution.[1]

The Landsat 8 Band designations highlight this difference in the spatial and spectral resolution of the Red, Green and Blue bands in the Visible region and the Panchromatic band.

Landsat 8 Operational Land Imager and Thermal Infrared Sensor (TIRS)

Bands	**Wavelength (micrometers)**	**Resolution (meters)**
	Visible	
Band 2 – Blue	0.452 – 0.512	30
Band 3 – Green	0.533 – 0.590	30
Band 4 - Green	0.636 – 0.673	30
	Panchromatic	
Band 8 – Panchromatic	0.503 – 0.676	15

Source: [2]

Image Fusion

Image Fusion[3] is defined as the process where we integrate a multitude of images over a single area of interest (AOI), with an intention of increasing the information inferred as compared to that obtained from the individual components accessed separately.

Image fusion can take place at various levels such as decision, feature and at a pixel level. As a result, we have a plethora of techniques with varying results. Hence, in order to compare them, a quantitative evaluation as well as a qualitative evaluation using some comparison metrics is important.

Pan Sharpening using PCA

Pan-sharpening is one of the techniques of Image fusion, which is defined as the merging of a multitude of images with an end-goal of achieving a greater degree of information in the resultant image.
In this tool, we present the user with a qualitative as well as a quantitative analysis of the process applied on the input images.

Some of the various Pan-Sharpening methods are :-

- Intensity – Hue – Saturation Transform based methods
  Suitable for a 3 band image where the separated intensity component is replaced by the Pan Image

- Principal Component Analysis based methods
  Principal component containing the largest variance is replaced by the Pan Image

- Brovey Method
  A pixel – by – pixel computation method related to the spectral angles

- Wavelet – based methods
  Involves transforming the image, combining the coefficient and a subsequent inverse transformation.

This technique that we deploy out of these is the method based on the PCT of the images. Using it, we replace the principle component of the multispectral image with an aim of retaining its spectral information and significantly increase the spatial resolution associated with it without loss of information.

The pan-sharpening process is heavily dependent on the co-registration of the two images provided to it as an input since the mapping is done at a pixel level. If data is not gathered from a satellite but from an aircraft or a drone, then the effectiveness of the process might be affected. Once we have obtained a multispectral image and a high spatial image falling within our area of interest, we then merge the two after transforming them into an alternative transform space. We subsequently calculate the significance of components based on the covariance of the data and generate the principal components using Matrix algebra. The merging process involves replacing the principal component obtained as a consequence of the aforementioned operations with the high spatial resolution image. After returning to the original space, all of the information is retained, moreover, the resultant image ends up being more interpretable than either of the original images.

Getting Started

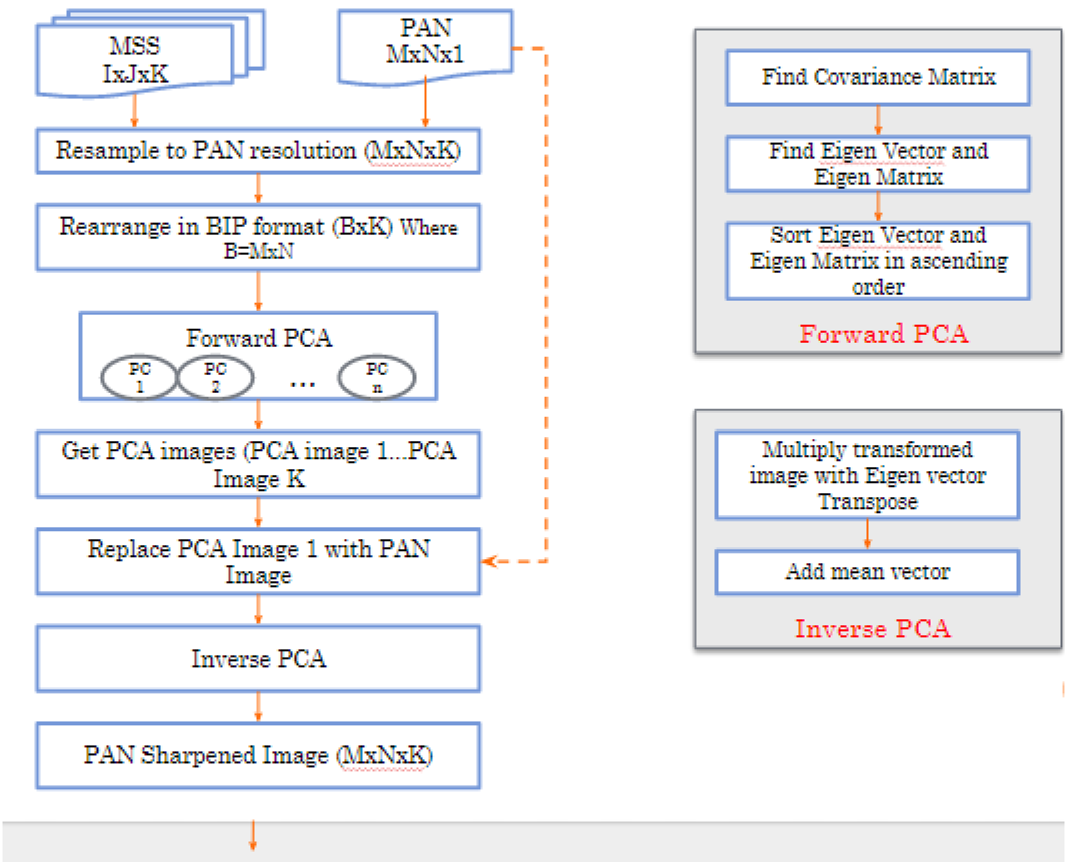
System Requirements

- A recent version of Windows, Linux or Mac OSX Operating System
- Personal computer or a notebook
- MATLAB 2015b (might not work as expected with other versions)

Quick Start

1. Copy all the files which are required for the proper functioning of the tool. This includes a figure file corresponding to the Graphical User Interface, Input images and several MATLAB scripts required for the execution of the code.
2. Open the MATLAB 2015b environment and navigate to the directory where the scripts are contained.
3. Open and run the file associated with the GUI named as pan_sharpening_using_pca.fig and run it.

Flow Diagram



Key Functions

pan_sharpening_using_pca.m

Callback: btn_getImage:-

```
function btn_getImage_Callback(hObject, eventdata, handles)
% hObject    handle to btn_getImage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.txt_ms_img_name,'string','');
[FileName,PathName] = uigetfile({'*.jpg;*.jpeg;*.tif;*.png;*.gif','All
Image Files'},'Select the Image file');
imagefilename = fullfile(PathName,FileName);
axes(handles.axes_image) ;


handles.ms_hs_image= double(imread(imagefilename));
handles.ms_image=  imresize(handles.ms_hs_image, 0.5, 'nearest') ;
imshow(uint8(handles.ms_image));

[r c d] =size(handles.ms_image);
str_size=strcat(num2str(r),'x',num2str(c),'x',num2str(d));
str_title=strcat({'Multispectral image of size'},{' '},{str_size});
set(handles.txt_ms_img_name,'string',str_title);

axes(handles.axes_interpolation);
[r c d] =size(handles.ms_hs_image);
str_size=strcat(num2str(r),'x',num2str(c),'x',num2str(d));
str_title=strcat({'Reference Image - High Spatial and Spectral  '},{'
'},{str_size});
set(handles.txt_interpolated_image,'string',str_title);
imshow(uint8(handles.ms_hs_image));


guidata(hObject,handles);
```

callback : Compute_pca:-

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

% [centred_matrix,mu,eigen_vector,
eigen_value,Image_pca1,Image_pca2,Image_pca3,pca] =
compute_pca(handles.three_band_image);
[centered_pan_image original_pca_image pca sharped ] =
perform_pansharp(handles.ms_image, handles.pan_image);

handles.centered_pan_image=centered_pan_image;
handles.original_pca_image=original_pca_image;
handles.pca=pca;
handles.sharped=sharped*255;

% display eigen vector and eigen values
set(handles.txt_eigen_value,'string',num2str(pca.lambda,3));
set(handles.txt_title_eigen_value,'string','Eigen Values');
set(handles.txt_eigen_vector,'string',num2str(pca.M,3));
set(handles.txt_title_eigen_vector,'string','Eigen Vectors');


% Find correlation between pan sharped image and High resolution
REference image
co_rel_red=corrcoef(handles.sharped(:,:,1),handles.ms_hs_image(:,:,1));
co_rel_green=corrcoef(handles.sharped(:,:,2),handles.ms_hs_image(:,:,2)
);
co_rel_blue=corrcoef(handles.sharped(:,:,3),handles.ms_hs_image(:,:,3))
;


% % display corelation
% set(handles.tit_corel,'string','Corelation between RGB bands of Pan
sharped Image and Orginal High Resolution Image');
%
% set(handles.txt_corel,'string',num2str(co_rel_red,2));
% set(handles.txt_cogreen,'string',num2str(co_rel_green,2));
% set(handles.txt_coblue,'string',num2str(co_rel_blue,2));

axes(handles.axes_pan_sharpened)

imshow(uint8(handles.sharped));
[r c d] =size(handles.sharped);
str_size=strcat(num2str(r),'x',num2str(c),'x',num2str(d));
str_title=strcat({'Pan sharped image of size'},{' '},{str_size});
set(handles.txt_sharpened_image,'string',str_title);

%compute SSIM
[ssimval, ssimmap] = get_ssim(handles.sharped,handles.ms_hs_image);

rmse = sqrt(immse(handles.ms_hs_image,handles.sharped));
```

```
set(handles.txt_ssim_val,'string',num2str(ssimval));
set(handles.txt_rmse,'string',num2str(rmse));

% figure;imshow(uint8(handles.ms_hs_image -
handles.sharped));title('difference');
% figure;imshow(uint8(handles.ms_hs_image));title('original');
% figure;imshow(uint8(handles.sharped));title('sharped');


guidata(hObject, handles);
```

callback:show_pca

```
% --- Executes on button p      ress in btn_show_pca.
function btn_show_pca_Callback(hObject, eventdata, handles)
% hObject    handle to btn_show_pca (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
pca1_img=handles.original_pca_image(:,:,1)*255;
pca2_img=handles.original_pca_image(:,:,2)*255;
pca3_img=handles.original_pca_image(:,:,3)*255;
setappdata(0,'pca1_img',pca1_img);
setappdata(0,'pca2_img',pca2_img);
setappdata(0,'pca3_img',pca3_img);
PCA_IMAGES;
```

perform_pansharp.m

```
function [centered_pan_image original_image PCA_parameters sharpened ]
= perform_pansharp( ip_ms_image, ip_pan_image )
%Image enhancement using Principal COmponent Merge Technique.
% First resize the multispectral image to same size as the Panchromatic
image by replicating values(nearest neighbour)
% Rearange the BSQ data to BIP Format
% Perform PCA transform
% Replace the first band with PCA image
% perform Inverse PCA to get PCA merged high resolution image
% Input parameters
%     ip_ms_image          -    Multispectral low spatial resolution
image to be transformed
```

```
%     ip_pan_image        -    Panchromatic High spatial resolution
image to be merged
%
% Output paramters
%      centered_pan_image    -  zero mean panchromatic image
%      original_image        -  Original PCA image without prominent
%                               component replaced by PAN image
%      PCAMap                - PCA parameters - mean, eigen value and
eigen
%                               vector
%      sharpened             -  PCA Sharpened Image
%
% Functions called
%     upsample_ms     - Resize the Multispectral image to size of
panchromatic image
%     multi_pca       - Perform the Principal Component Analysis
%
% % Coded by Harshula , Aarif, Ravi on 13/11/17

 ip_ms_image= double(ip_ms_image)/255;
 ip_pan_image = double(ip_pan_image)/255;

ip_ms_image= double(ip_ms_image);
ip_pan_image = double(ip_pan_image);

% ip_ms_image = upsample_ms(ip_ms_image,ip_pan_image);
[hr,hc]=size(ip_pan_image);
ip_ms_image = imresize(ip_ms_image,[hr hc]);

[m, n, d] = size(ip_ms_image);

ms_image = reshape(ip_ms_image, m*n, d);

%pca transform on ms bands
[PCA_Image, PCA_parameters] = multi_pca(ms_image, d);
PCA_Image = reshape(PCA_Image, [m, n, d]);
image_for_pca = PCA_Image;

 centered_pan_image = (ip_pan_image - mean(ip_pan_image(:))) *
std(image_for_pca(:))/std(double(ip_pan_image(:))) +
mean(image_for_pca(:));
 original_image = image_for_pca;
% %replace 1st band with pan
image_for_pca(:,:,1) = centered_pan_image;


%inverse PCA
image_for_pca = inv_pca(PCA_parameters.M, reshape(image_for_pca, m*n,
d), PCA_parameters.mean);

sharpened = reshape(image_for_pca, [m, n, d]);
```

upsample_ms.m

```
function [ high_res ] = upsample_ms( low_res,high_res )
% First resize the multispectral image to same size as the Panchromatic
image by replicating values(nearest neighbour)

[m, n, d] = size(low_res);
[hm, hn] = size(high_res);

high_res = zeros([hm, hm, d]);

for j = 1 : hm
    for k = 1 : hn
        high_res(j, k, :) = low_res( ceil(j/2), ceil(k/2), :);
    end
end

end
```

multi_pca.m

```
function [PCA_image, PCA_parameters] = multi_pca(ip_ms_image, no_dims)

% Perform PCA transform
% Replace the first band with PCA image
% perform Inverse PCA to get PCA merged high resolution image
% Input parameters
%     ip_ms_image          -    Multispectral low spatial resolution
image to be transformed
%
% Output paramters
%
%      PCA_image           -    PCA image
%      PCAMap              -    PCA parameters - mean, eigen value and
eigen

% Functions called
%     Compute_covariance            - Computes the covariance matrix
%     get_eigen_value_vector       - Returns the  eigen vector znd
eigen
%                                    values sorted in ascending order.
%
% % Coded by Harshula , Aarif, Ravi on 13/11/17



    if ~exist('no_dims', 'var')
        no_dims = 2;
    end

    % Make sure data is zero mean
    PCA_parameters.mean = mean(ip_ms_image, 1);
```

```
    ip_ms_image = ip_ms_image - repmat(PCA_parameters.mean,
[size(ip_ms_image, 1) 1]);

    % Compute covariance matrix
    if size(ip_ms_image, 2) < size(ip_ms_image, 1)
%          C = cov(X);
        C=compute_covariance(ip_ms_image);
    else
        C = (1 / size(ip_ms_image, 1)) * (ip_ms_image * ip_ms_image');
% if N>D, we better use this matrix for the eigendecomposition
    end

    % Perform eigendecomposition of C
    C(isnan(C)) = 0;
    C(isinf(C)) = 0;

%      %%%using std function eig
%      [eigen_vector, eigen_value] = eig(C);
%      % Sort eigenvectors in descending order
%      [eigen_value, ind] = sort(diag(eigen_value), 'descend');
%
%      if no_dims > size(eigen_vector, 2)
%          no_dims = size(eigen_vector, 2);
%          warning(['Target dimensionality reduced to ' num2str(no_dims)
'.']);
%      end
%    eigen_vector = eigen_vector(:,ind(1:no_dims));
%      eigen_value = eigen_value(1:no_dims);
%    %%%using std functio eig ends here

%get the sorted eigen value and correspondoing vectors
   [eigen_vector,eigen_value ] = get_eigen_value_vector( C );

    % Apply mapping on the data
    if ~(size(ip_ms_image, 2) < size(ip_ms_image, 1))
        eigen_vector = (ip_ms_image' * eigen_vector) .* repmat((1 ./
sqrt(size(ip_ms_image, 1) .* eigen_value))', [size(ip_ms_image, 2) 1]);
% normalize in order to get eigenvectors of covariance matrix
    end
    PCA_image = ip_ms_image * eigen_vector;

    % Store information for out-of-sample extension
    PCA_parameters.M = eigen_vector;
    PCA_parameters.lambda = eigen_value;
```

Sort_eigen_val_vector.m

```
function
[sorted_eigen_vector,sorted_eigen_val]=sort_eigen_val_vector(ip_eigen_v
ector,ip_eigen_val)
% this function takes in two matrices P and D, presumably the output
% from Matlab's eig function, and then sorts the columns of P to
% match the sorted columns of D (going from largest to smallest)
%
% EXAMPLE:
%
% ip_eigen_val =
```

```
%    -90     0     0
%      0   -30     0
%      0     0   -60
% ip_eigen_vector =
%      1     2     3
%      1     2     3
%      1     2     3
%
%
[sorted_eigen_vector,sorted_eigen_val]=sortem(ip_eigen_vector,ip_eigen_
val)
% sorted_eigen_vector =
%      2     3     1
%      2     3     1
%      2     3     1
% sorted_eigen_val =
%    -30     0     0
%      0   -60     0
%      0     0   -90


sorted_eigen_val=diag(sort(diag(ip_eigen_val),'descend')); % make
diagonal matrix out of sorted diagonal values of input D
[c, ind]=sort(diag(ip_eigen_val),'descend'); % store the indices of
which columns the sorted eigenvalues come from
sorted_eigen_vector=ip_eigen_vector(:,ind); % arrange the columns in
this order
```

get_eigen_val_vector.m

```
function [ sorted_eigen_vector,sorted_eigen_val ] =
get_eigen_value_vector( input_matrix )
%Computes the Eigen values and Eigen vectors for the given matrix
%Input
%   ip_matrix- BIP format input matrix whose covariance is to be
calculated
%output
%   sorted_eigen_vector     - Eigen vector of the input matrix  in
descending
%                             order of eigen values
%   sorted_eigen_val        - Descending order eigen_value

% % Coded by Harshula , Aarif, Ravi on 13/11/17


% [eigen_vector1,eigen_val1]=eig(input_matrix);
% disp('Eigen vector and Eigen values are displayed in Descending order
of Eigen value');
% disp('Sorted Eigen vector and Eigen values using Built in Matlab
function');
%
[sorted_eigen_vector1,sorted_eigen_val1]=sort_eigen_val_vector(eigen_ve
ctor1,eigen_val1)
```

```
error_margin = 1e-16;
[eigen_vector,eigen_val] =
compute_eigen_val_vectors(input_matrix,error_margin,0);
[r c]=size(eigen_vector);

for i = 1:c
  if (eigen_vector(:,1)<0)
    eigen_vector(:,1)=eigen_vector(:,1)*(-1);
  end
end


%disp('Sorted Eigen vector and Eigen values using developed function');

[sorted_eigen_vector,sorted_eigen_val]=sort_eigen_val_vector(eigen_vect
or,eigen_val);


end
```

ssim_fig.m

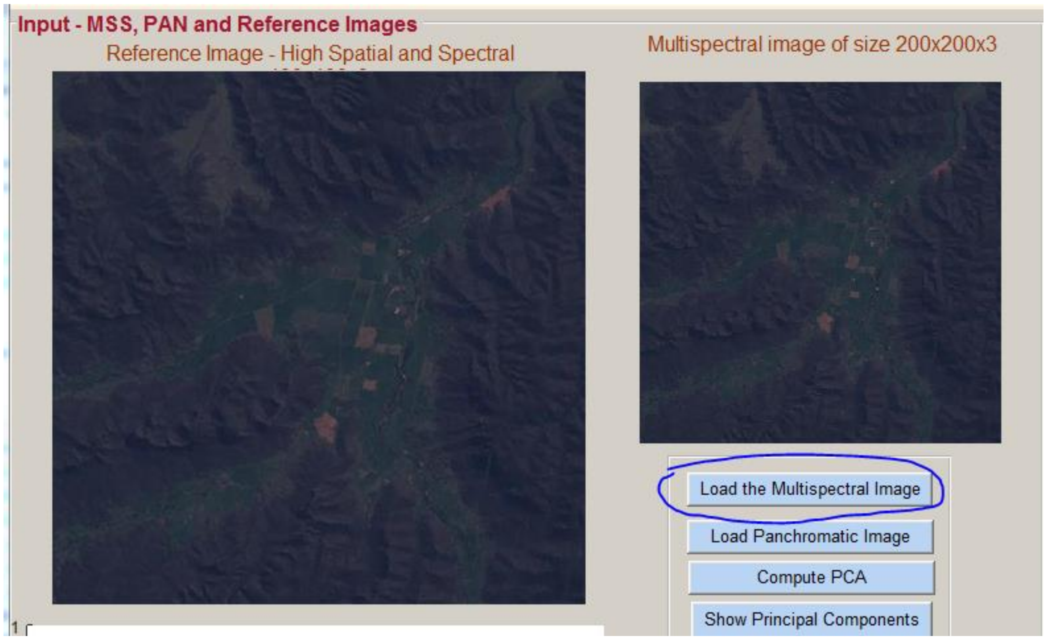
```
% Get default command line output from handles structure
ssimval = getappdata(0,'ssim');
ssimmap = getappdata(0,'ssimmap');
diff_image = getappdata(0,'diff_image');
rmse = getappdata(0,'rmse');

axes(handles.axes_ssim);
imshow(ssimmap,[]);
title(sprintf('SSIM Index Map - Mean ssim Value is %0.2f',ssimval));

axes(handles.axes_diff);
imshow(diff_image,[]);

set(handles.txt_rmse,'string',num2str(rmse));
```

PCA_IMAGES.m

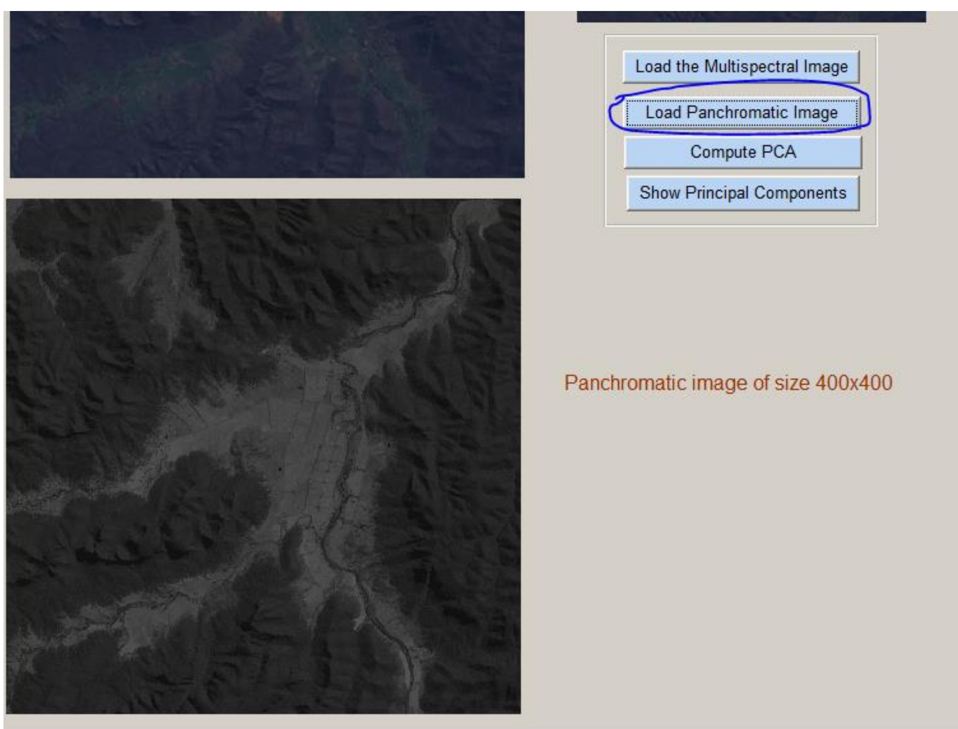
```
axes(handles.axes_pca1);
imshow(pca1_img,[]);
axes(handles.axes_pca2);
imshow(pca2_img,[]);
axes(handles.axes_pca3);
imshow(pca3_img,[]);
```

Step by Step Execution

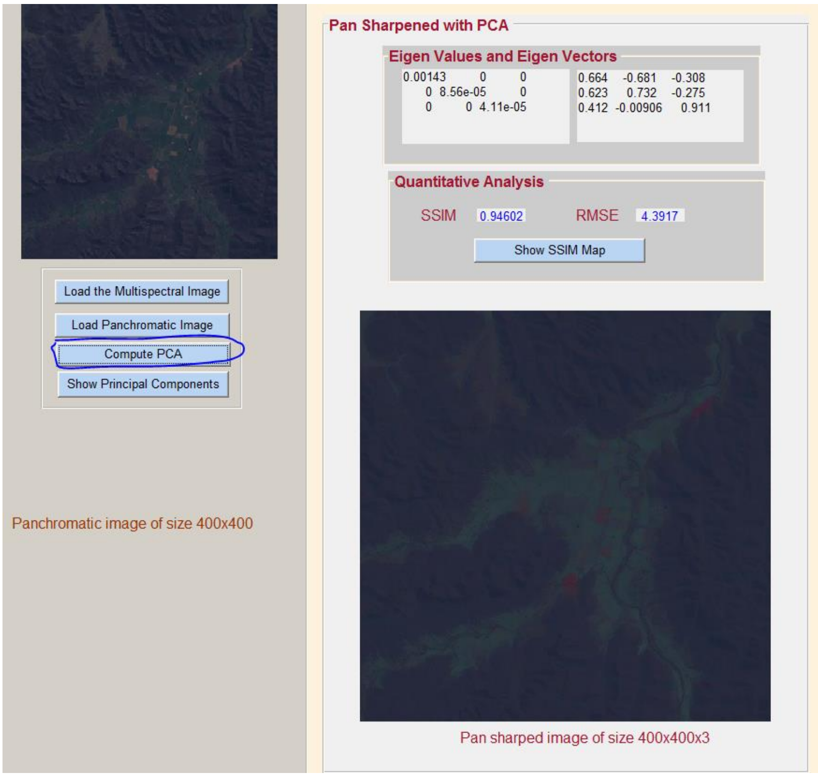
Step 1 Load the Multispectral images



Step 2 Load the Panchromatic Image



Step 3 Compute PCA



Remarks

- PCA algorithm is dependant on scene content – high vegetation content causes poor performance.
- High NIR contribution tends to distort the PCA transformation and cause blurriness.
- Sharpness is an issue, especially in scenes where there is a lot of green vegetation.
- PCA algorithm doesn’t seem to handle a variety of scene content (works best in urban settings)
- PCA based pan-sharpening considers the overall variance of the image and not the local variance

Quantitative Analysis

Some of the quality indices commonly used to measure the accuracy of fused images with respect to the original image are :-

- MSE and RMSe
- SSIM
- SAM
- Correlation Coefficient
- Relative Dimensionless Global Error in Synthesis
- Universal Image Quality Index
- Quaternions Theory Based Quality Index

The indices we use are the SSIM and The RMSE.

- SSIM is a measure used to measure similarity between 2 images
- RMSE – It is a good indicator of the spectral quality of the fused image. The value 0 indicates that best spectral information was retained in the fused image.

- SSIM

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x + \mu_y + C_1)(\sigma_x + \sigma_y + C_2)}$$

where

$$\mu_x = \sum_{i=1}^{N} \omega_i x_i$$

$$\sigma_x = \left(\sum_{i=1}^{N} \omega_i (x_i - \mu_x) \right)^{\frac{1}{2}}$$

$$\sigma_{xy} = \sum_{i=1}^{N} \omega_i (x_i - \mu_x)(y_i - \mu_y)$$

The constants C_1 and C_2 are defined according to the following expressions:

$$C_1 = (K_1 L)^2$$
$$C_2 = (K_2 L)^2$$

- RMSE

$$\sqrt{\frac{\sum_{i=1}^{n} (x_i - y_i)^2}{n}}$$

References

1. [Process for enhancing the spatial resolution of multispectral imagery using pan-sharpening]
2. [Band Designations for various Landsat Satellites]
3. [On the performance evaluation of Pan-Sharpening Techniques]