Image fusion techniques for Remote Sensing Optical Payloads – Case study on Sentinel Imagery and Visualization using OSG Earth

A

Project Report

Submitted in partial fulfilment of the

Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

Vanteddu Akhila (1602-18-737-063)

Kota Sai Vishwanath Somana (1602-18-737-077)

Sandali Nemmaniwar (1602-18-737-103)

Syed Ayaz Moinuddin (1602-18-737-111)

Under the guidance of

D.R.L. Prasanna

Assistant Professor

Department of Information Technology

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31 2022

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Information Technology

DECLARATION BY THE CANDIDATE

We, **Vanteddu Akhila, Kota Sai Vishwanath Somana, Sandali Nemmaniwar** and **Syed Ayaz Moinuddin** bearing hall ticket number, **1602-18-737-063**, **1602-18-737-077, 1602-18-737-103, 1602-18-737-111** hereby declare that the project report entitled **Image fusion techniques for Remote Sensing Optical Payloads – Case study on Sentinel Imagery and Visualization using OSG Earth** under the guidance of **D.R.L. Prasanna,** Assistant Professor, Department of Information Technology, Vasavi College of Engineering, Hyderabad, is submitted in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering** in **Information Technology**

This is a record of bonafide work carried out by us and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

Vanteddu Akhila
1602-18-737-063
Kota Sai Vishwanath Somana
1602-18-737-077
Sandali Nemmaniwar
1602-18-737-103
Syed Ayaz Moinuddin
1602-18-737-111

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Information Technology

BONAFIDE CERTIFICATE

This is to certify that the project entitled **Image fusion techniques for Remote Sensing Optical Payloads – Case study on Sentinel Imagery and Visualization using OSG Earth** being submitted by **Vanteddu Akhila, Kota Sai Vishwanath Somana, Sandali Nemmaniwar, Syed Ayaz Moinuddin** bearing **1602-18-737-063, 1602-18-737-077, 1602-18-737-103, 1602-18-737-111** in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Information Technology is a record of bonafide work carried out by them under my guidance.

D.R.L. Prasanna	**Dr. K. Ram Mohan Rao**
Assistant Professor	**HOD, IT**
Internal Guide	

External Examiner

ACKNOWLEDGEMENT

The satisfaction that accompanies that the successful completion of the project would not have been possible without the kind support and help of many individuals. We would like to extend our sincere thanks to all of them. We would like to take the opportunity to express our humble gratitude to **Dr. K. Raghavendra**, Scientist 'SG' at ADRIN and **D.R.L. Prasanna**, Assistant Professor under whom we executed this project. We are also grateful to **S.K. Chaya Devi**, Associate Professor for her guidance. Their constant guidance and willingness to share their vast knowledge made us understand this project and its manifestations in great depths and helped us to complete the assigned tasks. We would like to thank all faculty members and staff of the Department of Information Technology for their generous help in various ways for the completion of this project. Finally, yet importantly, we would like to express our heartfelt thanks to our Head of Department **Dr. K. Ram Mohan Rao.**

ABSTRACT

Image fusion refers to the process of combining two or more images into single image, integration/composition of the all such image information into a single composite image which is useful for various applications like image interpretation and further studies. The resultant image has higher information content compared to such individual input images being used for fusion.

The goal of the fusion process is to evaluate the information at each pixel location in the input images and retain the information from that image which best represents the true scene content or enhances the utility of the fused image for a particular application. Image fusion is a vast discipline in itself, and refers to the fusion of various types of imagery that provide complementary information.

In this project work, we propose to study image fusion techniques for satellite remote sensing optical payloads. Due to proliferent information contained in the satellite imagery with temporal resolutions could be composed to create a fused image for better analysis and study applications. Here, we perform such study using machine learning based image fusion techniques on Sentinel imagery using, Principal Component Analysis and Wavelet based image fusion. We focus on the datasets of the Sentinel-2(2 satellites), which is part of Europe's ambitious Copernicus program.

We take images of an area that are captured by the Sentinel-2 satellites into consideration and fuse the pixel values of the images in order to retrieve the missing part(s) in the images or make those parts more visible. After the fused image is produced, we perform statistical analysis on the input images and the output images. This analysis is done by calculating mean squared error and structural similarity index between the images. These two statistics are used to compare images. We incorporate these statistics to compare our input and output images. We will further perform visualization using OSG Earth. The main focus of visualization is to give an insight to the area of fused image.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1 Overview

1.1.1 Sentinel data

Copernicus is the European Union's Earth observation programme coordinated and managed by the European Union Agency for the Space Programme in partnership with the European Space Agency (ESA). The main aims of this programme are achieving a global, continuous, autonomous, high quality, wide range Earth observation capacity. This programme monitors the atmosphere, marine, land, climate, emergency and security themes of our planet. It consists of sentinel missions; each mission is a constellation of two satellites.

Sentinel satellites cover from visible to short-wave infrared regions in the spectrum and the spatial resolution is 5 to 60m range. It's revisit time ranges from two days to a week, depending upon the location. Thus, they have huge impact on applications such as map updating, flood monitoring, weather forecasting etc., The data and information produced in Copernicus framework are made available free-of-charge to all its users and the public. Now, this initiative has given an opportunity to use this data which is a valuable set of information as tools for many remote sensing and earth monitoring applications.

In particular, the sentinel-2(S2) satellite mission consists of two satellites: sentinel-2A and sentinel-2B. This provides high resolution imagery for land services. It has high revisit time, 10 days at the equator with one satellite, and 5 days with 2 satellites under cloud-free conditions which results in 2-3 days at mid-latitudes. The data provided by S2 depends on the spectral reflectivity of the target illuminated by sunlight. S2 has 13 bands, which can be seen in table (1.1).

Sentinel-2 Bands	Central Wavelength (μm)	Resolution (m)
Band 1 – Coastal aerosol	0.443	60
Band 2 – Blue	0.490	10
Band 3 – Green	0.560	10
Band 4 – Red	0.665	10
Band 5 – Vegetation Red Edge	0.705	20
Band 6 –Vegetation Red Edge	0.740	20

Band 7 – Vegetation Red Edge	0.783	20
Band 8 – NIR	0.842	10
Band 8A – Vegetation Red Edge	0.865	20
Band 9 – Water vapour	0.945	60
Band 10 – SWIR - cirrus	1.375	60
Band 11 – SWIR	1.610	20
Band 12 – SWIR	2.190	20

Table 1.1 Sentinel-2 bands

There are few limitations on S2 imagery such as, cloud coverage limits the view of a particular area. Another one being the low resolution of imagery. The 10m resolution would be helpful in detecting large changes in the infrastructure of an area or identifying any huge changes in environment but not very helpful in detecting smaller changes. This is because of the level of detailing in S2 images is not very high. In this project work, we focus on enhancing the features of the images that are acquired by S2. When we say enhancing, we mean extracting the missing features that are lost due to clouds or noise and increase the level of detail in the images. Image fusion, which is image processing algorithms, are used for our work.

1.1.2 Image fusion

Image Fusion is used to retrieve important data from a set of input images and put it into a single output image to make it more informative and useful than any of the input images. In this way, the quality and application range of the data improves. The goal of the fusion process is to evaluate the information at each pixel location in the input images and retain the information from that image which best represents the true scene content or enhances the utility of the fused image for a particular application. Fig (1.1) shows the architecture of image fusion technique, where it is shown that two images are taken to consideration, on which a fusion rule is applied to get a fused image. Fusion rule is a sort of constraint or a set of rules for the pixel, based on which fusion process takes place. The most common and simple fusion rules are average rule, in which fusion is conducted by averaging the pixel values of the images, minimum rule, in which the minimum of two pixel values is used to produce fused image and maximum rule, which is similar to minimum rule, but maximum pixel value is taken into consideration.

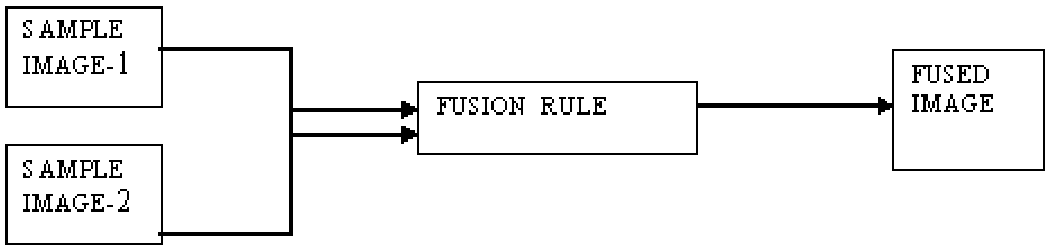


Fig 1.1 Image fusion Architecture

There can be different set of classifications when it comes to image fusion techniques. The main ones can be pixel-based methods, feature-based methods and decision-based methods. These are general methods, but in remote sensing context we can classify image fusion techniques into multi-resolution, multi-temporal, multi-sensor and mixed. In multi-resolution image fusion, only a single sensor with different resolution bands is considered. In multi-temporal, fusion of images obtained from same sensor obtained at different times is involved. Multi-sensor technique, as the name suggests involves fusing images from different sensors together. Mixed image fusion technique could be combination of any of the mentioned techniques. For example, hyper spectral and multi-resolution images can be fused together to get spatial spectral full resolution image or data cube. Image fusion techniques can also be classified as spatial domain techniques and frequency domain techniques depending on the domain in which the fusion is carried out. The overview of these techniques is mentioned in fig (1.2).

In the context of remote sensing, we have seen few categories that image fusion is classified into. Sensors with high spectral resolution, do not have an optimal spatial resolution, which leads to a poor performance went it comes to feature identification tasks. On a high spatial resolution panchromatic image (PAN), detailed features can be recognized easily, but the spectral information is not as much in these images. For multispectral (MS) images, it is the opposite. The qualities and contents of the image can be of better use if the high spatial and spectral resolutions are integrated into one image. As the level of detailing is high in integrated image, the area of applications is also widened.

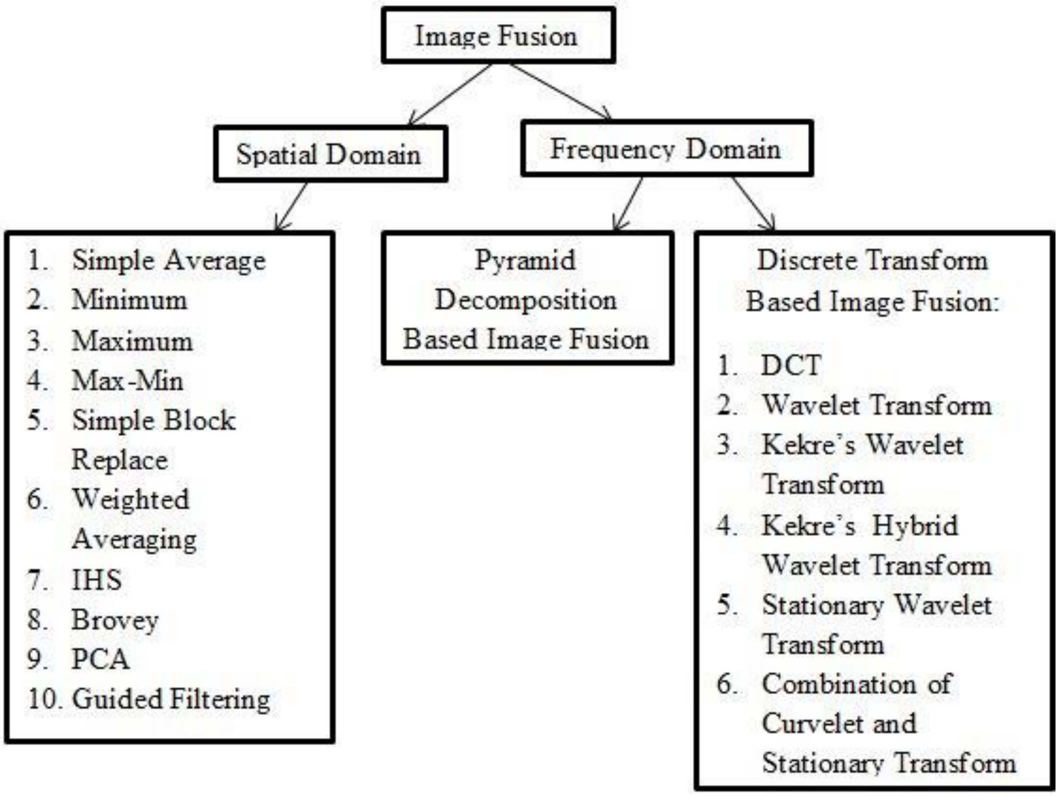


Image Fusion
Spatial Domain
Frequency Domain
1. Simple Average
2. Minimum
3. Maximum
4. Max-Min
5. Simple Block Replace
6. Weighted Averaging
7. IHS
8. Brovey
9. PCA
10. Guided Filtering
Pyramid Decomposition Based Image Fusion
Discrete Transform Based Image Fusion:
1. DCT
2. Wavelet Transform
3. Kekre's Wavelet Transform
4. Kekre's Hybrid Wavelet Transform
5. Stationary Wavelet Transform
6. Combination of Curvelet and Stationary Transform

Fig 1.2 Domain specific fusion techniques

With proper algorithms, like the ones present in the fig (1.2), it is possible to combine the MS and PAN images into single image with better features. The integrated image should contain the highest possible spatial information content while still preserving good spectral information quality. The result of image fusion is a new image which is more suitable for human and machine perception or further image-processing tasks such as segmentation, feature extraction and object recognition.

1.2 Proposed Method

Firstly, we collect the datasets of the Sentinel-2 satellite mission. Image fusion is performed on this dataset i.e, we combine two or more images of same area at different temporal sequences into one composite to extract missing features. The result is these images has higher information content compared to any of the input images from the dataset. Later to validate the fusion results, we perform statistical analysis on the images.

Fusion is performed using both the PCA and wavelet-based image fusion techniques. With the help of statistical analysis, we will analyze which method is more suitable for the sentinel satellite images. Mean squared error and structural similarity index are the methods that we use under statistical analysis. And finally, visualization is performed in 3-dimensional form using the OSG tool.

2. LITERATURE SURVEY

There are researchers that have performed image fusion with different approaches. We have studied few existing work related to image fusion, to go ahead with our projects and they are mentioned underneath.

Scarpa, Giuseppe & Gargiulo, Massimiliano & Mazza, Antonio & Gaetano, Raffaele [1] proposed a work related to image fusion. The objective of their work is to propose and test a set of solutions to estimate a target optical feature at a given date from images acquired at adjacent dates, or even from the temporally-closest SAR image.
Under cloudy conditions, however, optical-based features are not available, and they are commonly reconstructed through linear interpolation between data available at temporally-close time instants. In this work, we propose to estimate missing optical features through data fusion and deep-learning. Several sources of information are taken into account—optical sequences, SAR sequences, digital elevation model—so as to exploit both temporal and cross-sensor dependencies. Based on these data and a tiny cloud-free fraction of the target image, a compact convolutional neural network (CNN) is trained to perform the desired estimation.

Experimental results of this work have been very promising, showing a significant gain over baseline methods according to all performance indicators. Besides proving the potential of deep learning for remote sensing, experiments have shown that SAR images can be used to obtain a meaningful estimate of spectral indexes when other sources of information are not available. The only limitation is, this requires the creation of a large representative dataset for training.

Metwalli, Mohamed & Nasr, A.H. & Faragallah, Osama & El-Rabaie, El-Sayed [2] proposed a method to integrate between the two families PCA and HPF to provide pan sharpened image with superior spatial resolution and less spectral distortion. This procedure has been assessed on two types of remote sensing data with different spatial and spectral properties.

An integrated PCA and HPF fusion approach is proposed. The resampled multi-spectral image is transformed with PCA. The panchromatic image is smoothed by a Gaussian filter. The high spatial detail of panchromatic image is extracted as the difference between the original panchromatic image and the smoothed one. A linear combination of the extracted high spatial detail of the panchromatic image into the first principal component using a gain parameter as the ratio of standard deviation of pc1 to standard deviation of panchromatic image. The new first principal component and other principal component are transformed with the inverse PCA to obtain the pan sharpened multi-spectral image.

This method has significantly reduced the spectral distortion compared with the PCA, HPF, and GS fusion methods. The spatial quality of the proposed fusion method is higher than the HPF fusion method and comparable to the PCA and GS fusion methods.

Krista Amolins*, Yun Zhang, Peter Dare [3] proposed a method in which, an introduction to wavelet transform theory and an overview of image fusion technique are given, and the results from a number of wavelet-based image fusion schemes are compared. It has been found that, in general, wavelet-based schemes perform better than standard schemes, particularly in terms of minimizing color distortion. Schemes that combine standard methods with wavelet transforms produce superior results than either standard methods or simple wavelet-based methods alone.

The results from wavelet-based methods can also be improved by applying more sophisticated models for injecting detail information. The drawback is that there is greater computational complexity and often parameters must be set up before the fusion scheme can be applied.

Kaur, H., Koundal, D. & Kadyan, V [4] proposed a survey on image fusion techniques is done. The work shows, how different image fusion techniques work and their processing is analysed and further discussed.

Averaging, minimum pixel value, simple block replacement, maximum pixel value, max- min are very easy to implement. weighted averaging, improve the detection reliability. Laplacian/gaussian pyramid, low pass pyramid ratio, morphological pyramid, gradient pyramid, filter subtract decimate, provide a better image quality of a representation for multi focus images. The CNN is able to extract features and representation can learn most elective features from training data without any human intervention. CSR compute sparse representation of an entire image shift-invariant representation approach elective in details preservation less sensitive to mis-registration.

Few limitations of this methodology are that, the averaging, minimum pixel value, simple block replacement, maximum pixel value, max- min, decreases the image quality, reduces noise into final fused resultant image and produce blurred images. Not appropriate for real time applications. CNN has high computational cost and CSR need a lot of training data.

Size Li, Pengjiang Qian, Xin Zhang, Aiguo Chen [5] proposed a method in which denoising of image is performed and statistics are used.

Basic Theory of Residual Network: Residual network consists of a stack of residual blocks, all of which can be represented in a general form. We assume that H(x) is the basic mapping of several stacked layers, and x represents the input of the first layer network can be designed with a jump connection, i.e., H(x) = F(x) + x, which means to add the two input elements. Using the jump connection, the gradient-related problems of the deep neural network can be solved without increasing the computational complexity. In addition, an approximate residual function, i.e., F(x) = H(x) − x, can be obtained with the gradual approximation of multiple nonlinear layers.

PSNR is often used for objectively evaluating image quality [15]. It represents the ratio of the maximum power of the effective signal to the noise power in the signal. .e unit of PSNR is dB, and the mathematical expression of PSNR.

$$\mathrm{PSNR} = 10 \times \log_{10}\left(\frac{(2^n - 1)^2}{\mathrm{MSE}}\right),$$

where MSE denotes the mean squared error between the original image and the generated image,

$$\mathrm{MSE} = \frac{1}{mn}\sum_{i=1}^{n}\sum_{j=1}^{m}\left\|S_{ij} - T_{ij}\right\|^2,$$

SSIM is another indicator to quantify the image quality [16]. It is adopted in this paper since PSNR cannot accurately indicate the clarity of the image in some cases e value of SSIM ranges from −1 to 1. Specifically, SSIM =1 means that the structures of the two images are completely the same. On the contrary, the structures of the two images are completely different, if SSIM is equal to −1. SSIM is given by,

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)},$$

For denoising method based on convolutional neural network, no pairwise training samples are needed, which overcomes the problem of insufficient pairwise training samples in real images. In denoising method based on residual network, problems of gradient disappearing and gradient explosion are solved effectively, and the convergence speed is accelerated. In case of denoising method based on Generative Adversarial Network, it can generate realistic noise images, expand the real image dataset, and solve the problem of insufficient training samples. And for denoising method based on Graph Neural Network, complex noise distribution can be well fitted by the topology of graph network.

The limitations in denoising method based on convolutional neural network is that, shallow pixel level information utilization is low and texture details are easily lost. In denoising method based on Generative Adversarial Network, there are some problems such as unstable network training, slow convergence speed, and uncontrollable model.

Andrea Garzelli [6] has proposed a review of image fusion algorithms based on the super-resolution paradigm**.** In this paper they worked on different approaches

- Restoration-Based Approaches
- Sparse representation
- Bayesian approaches
- Variational approaches

They compared 4 SR Techniques with the AWLP algorithm and compared their performance with the AWLP algorithm. Sparse FI and J-Sparse FI are the only methods that provide confident quality improvement over AWLP.

For the most common application fields in the broad domain of remote sensing image fusion, the algorithms based on the super-resolution paradigm are not yet mature for solving fusion processing tasks in operational remote sensing system.

Another drawback of these new approaches to remote sensing image fusion is their extremely high computational complexity.

Qeethara Al-Shayea, Muzhir Al-Ani [7] proposed an algorithm of image visualization is used to generate 3D objects depending on the extraction of the important features. This method provides high level resolution images with minimum noise as resultant images.

- Implement noise reduction.
- Implement edge enhancement.
- Calculate surface rendering.
- Calculate volume estimation

Ganesan Gunasekaran and Meenakshisundaram Venkatesan [8] proposed an efficient technique for three-dimensional image visualization through two-dimensional images. The algorithm is**,** in the first stage, the input images are subject to pre-processing as they may have noises. In the second stage in this proposed method, various filtering techniques, such as de-noising, decimation, and multiresolution and mesh generation filters, are employed. The third stage deals with edge enhancement of noise-removed input images. Basically, edge enhancement is the technique of digital image processing filter that improves the edge similarities of given images in an effort to enhance the sharpness of the images. The fourth stage is focused on the rendering process, and volume rendering methods have been established to overcome the issues of the perfection of illustration of surfaces in the iso-surface methods.

3. SYSTEM REQUIREMENTS AND SPECIFICATIONS

Here, we describe software, hardware followed by user requirements in the realization of the system.

3.1 Software requirements

Software requirements can be described as the features and functionalities that are expected from the project's end product. We need these requirements to understand how the project is proceeding into further steps. The key software requirements for this project are:

- Jupyter Notebook
- Anaconda navigator
- OSG Earth
- OpenSceneGraph
- QGIS Software

The purpose of each software requirement is elaborately discussed in the following sections.

3.1.1 Jupyter Notebook:

The Jupyter Notebook App is a server-client application that allows editing and running via a web notebook documents browser. It can be executed on a local desktop requiring no internet access or can be installed on a remote server and accessed through the internet.

Notebook documents are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements. Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc..) as well as executable documents which can be run to perform data analysis. A notebook kernel is a "computational engine" that executes the code contained in a Notebook document.

In this project work we use the Jupyter Notebook app to run the image fusion codes. As our codes are written in python language, the ipython kernel executes the codes.

3.1.2 Anaconda Navigator:

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository. It is available for Windows, macOS, and Linux.

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly. Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.

We need OS module, PyWavelets module, OpenCV library, NumPy package, Python Imaging LibraryMatplotlib package, Scikit-image package for our project. They are installed with the help of Anaconda Navigator.

3.1.3 OSGEarth:

OSGEarth is a geospatial SDK and terrain engine for OpenSceneGraph applications. The goals of osgEarth are to:

- Enable the development of 3D geospatial appliations on top of OpenSceneGraph.
- Make it as easy as possible to visualize terrian models and imagery directly from source data.
- Interoperate with open mapping standards, technologies, and data.
- Since osgEarth is a free open source SDK, the source code is available to anyone and we welcome and encourage community participation when it comes to testing, adding features, and fixing bugs.

Through this we are visualizing the map data by adding terrain data and elevation data so that we can view the 3-Dimensional view of the buildings.

3.1.4 OpenSceneGraph:

The OpenSceneGraph is an open source high performance 3D graphics toolkit, used by application developers in fields such as visual simulation, games, virtual reality, scientific visualization and modelling. Written entirely in Standard C++ and OpenGL it runs on all Windows platforms, OSX, GNU/Linux, IRIX, Solaris, HP-Ux, AIX and FreeBSD operating systems. The OpenSceneGraph is now well established as the world leading scene graph technology, used widely in the vis-sim, space, scientific, oil-gas, games and virtual reality industries.

Inorder to work with osgearth first we need to install openscenegraph. Osgearth files uses the openscenegraph libraries in order to display the contents of the earth files.

3.1.5 QGIS Software:

Quantum GIS (QGIS) is an open source Geographic Information System that supports most geospatial vector and raster file types and database formats. The program offers standard GIS functionality, with a variety of mapping features and data editing. QGIS also has support for plugins that expands its functionality further by providing additional tools namely: GPS data support, geo referencing, and additional mapping components.

QGIS is one of the most popular software for working with spatial data. It has flexibility to scale with user requirements: from a simple data viewer, to data collection, editing and analysis, to serving data on the web – on as numerous machines as required and without any licensing concerns. As we are working with sentinel imagery, we require QGIS software to view the input and output images. This will help us understand the fused features in the fused image better.

3.2 Hardware Requirements

As we are fusing images, high processing is required. For this we should use GPU. Although a minimum of 8GB RAM can do the job, 16GB RAM and above is recommended for these tasks. When it comes to CPU, a minimum of 7th generation (Intel Core i5 processor) is recommended.

3.2.1 GPU:

The graphics processing unit, or GPU, has become one of the most important types of computing technology, both for personal and business computing. Designed for parallel processing, the GPU is used in a wide range of applications, including graphics and video rendering. Although they're best known for their capabilities in gaming, GPUs are becoming more popular for use in creative production and artificial intelligence (AI).

3.3 USER REQUIREMENTS

The system will take sentinel imagery as input. The system is fed with two images from the sentinel-2 data set at a time. The output will be the fused image which will contain the best features of both the input images. We form statistical analysis on the input and output images to analyze which image fusion technique is working better. And later, in visualization, an earth file is created by adding image layer first where it will directly take mapbox satellite data and further add elevation data and we provide styling for the buildings roads, rivers, trees through CSS and finally we run the application with the osgearth_viewer tool to view the map.

4. METHODOLOGY

4.1 UML diagrams:

4.1.1 Use-case diagram:

Use case diagrams are the diagrammatic representations depicting users' interactions with the system. Fig (4.1) refers to the use case diagram of our project work.

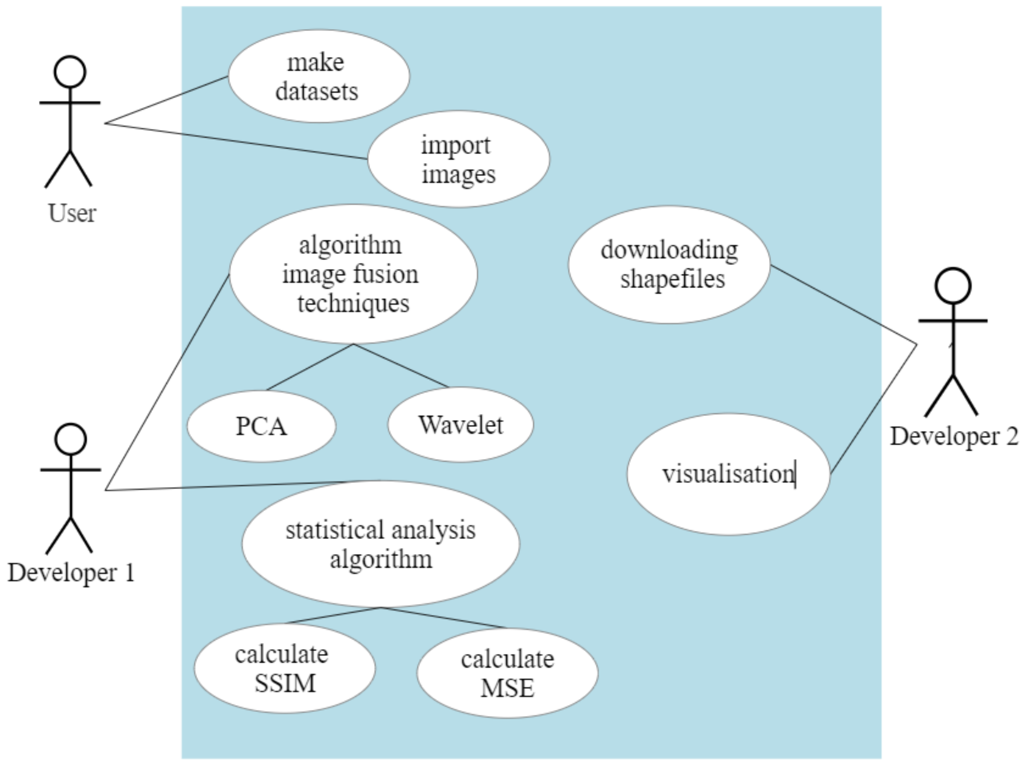


Fig 4.1 Use-case diagram

The project’s use case diagram shows what happens in the system. There are three actors working on the system with different functions for each. The first one is the user. User creates dataset and imports the images. The next actor is the developer who works on the fusion part of the project. The developer applies fusion algorithms which are PCA and wavelet-based techniques. This developer also performs statistical analysis using two methods which are mean squared error and structural similarity index method. The third actor is the developer who works on the visualization part. This actor downloads the shapefiles and performs visualization using certain applications.

4.1.2 Sequence diagram:

A sequence diagram depicts the sequence of the messages between the objects. It helps clearly show the requirements of the application removing the design part. Fig (4.2) is the sequence diagram that depicts the image fusion part.

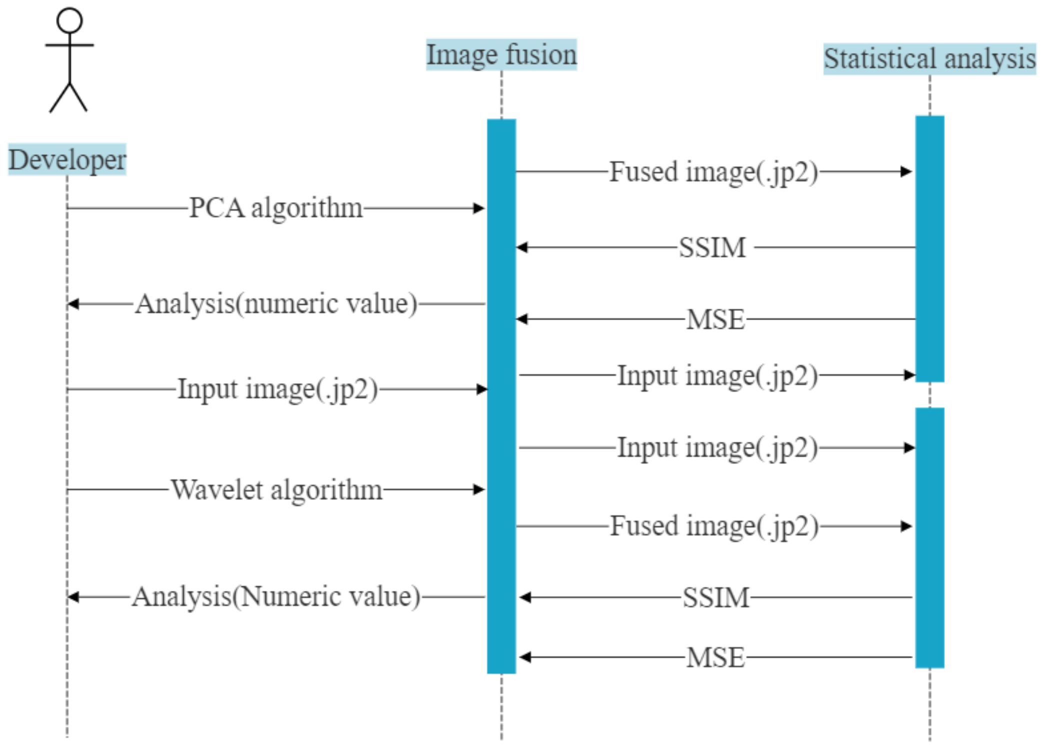


Fig 4.2 Sequence diagram for Image Fusion

There are 3 objects in the sequence diagram of our project. They are Developer, Image fusion and Statistical analysis. The developer writes the PCA and wavelet algorithms and gives the input images to make this algorithm work. Next in the image fusion, these two techniques are run as a code and a fused image is produced as an output from both the algorithms each. Along with this fused image the input image is also fed to the system for performing statistical analysis. The analysis is performed using 2 methods and a numeric value is given as the output for the developer to study the results of the fusion techniques.

Fig (4.3), describes the sequence of events happening in visualization. First the developer will download necessary files required for visualizing. Then with the downloaded data the developer will generate an earth file for performing visualization. Finally, the developer can view the output in 3-D form.

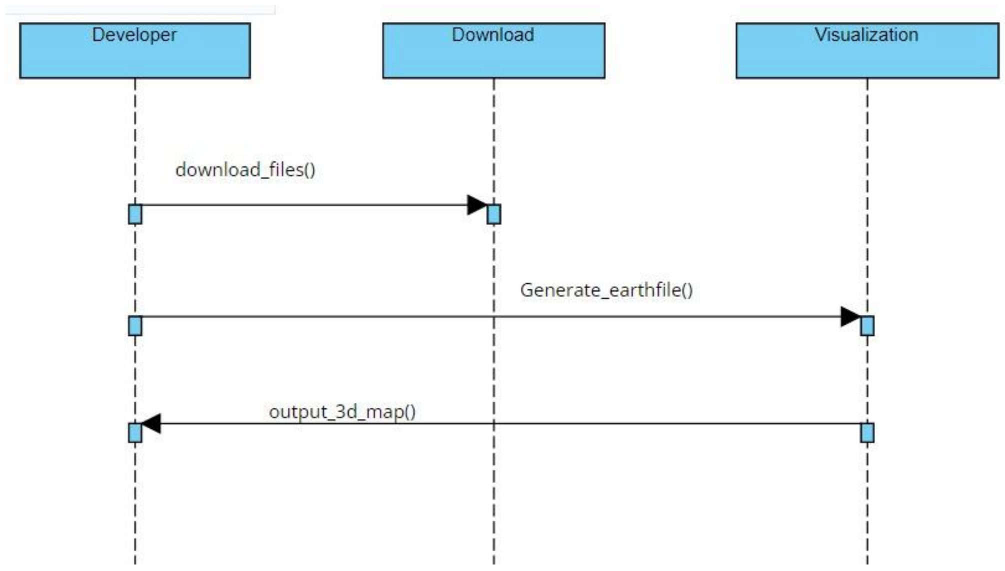


Fig 4.3 Sequence diagram for Visualization

4.1.3 Data-flow diagram:

A dataflow diagram is a diagram where the flow of data is showed using a set of processes. It also provides information about the outputs and the inputs of each entity in the process. Fig (4.4) is the data flow diagram of image fusion process.

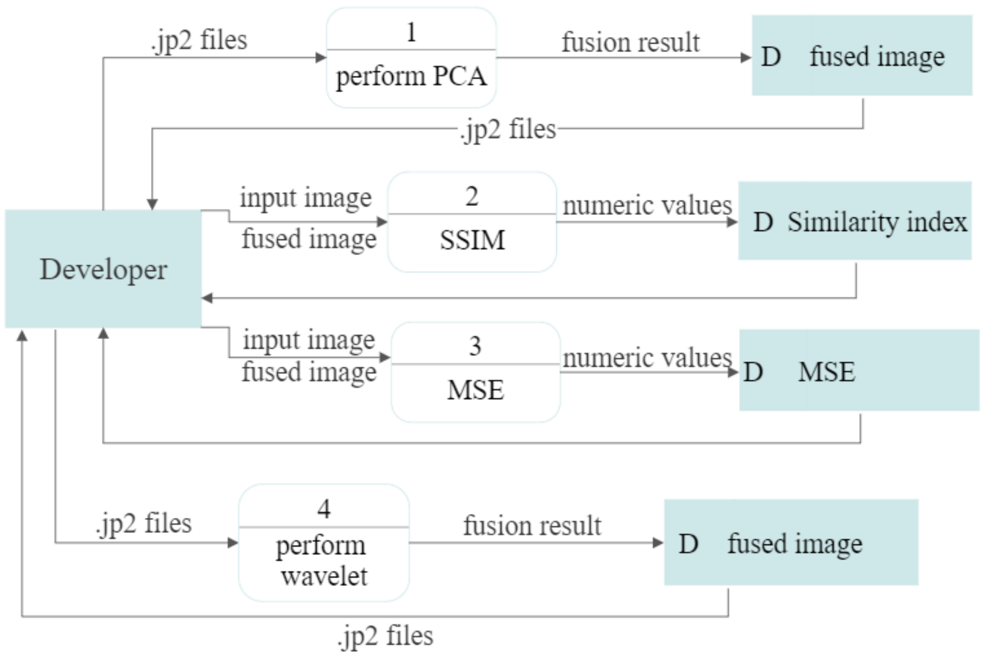


Fig 4.4 Data-flow diagram for Image Fusion

The object in the diagram is the Developer. Through the developer input image is given to the process 1 called perform PCA in jp2 format. Similarly, input image is also given to the process 4 perform wavelet in jp2 format. After the techniques are performed a fused image is produced and the image from both the methods is in jp2 format. This output data is observed by the developer. Next the developer performs statistical analysis using two processes 2,3 called SSIM and MSE respectively. The data fed to these two processes is in the form of jp2 image. After the analysis is the done the system shows the result as some numeric value which is returned back in the same format to the developer.

Fig (4.4) refers to the flow of data, first developer downloads the files and perform visualization then developer gets an 3D model as an output.

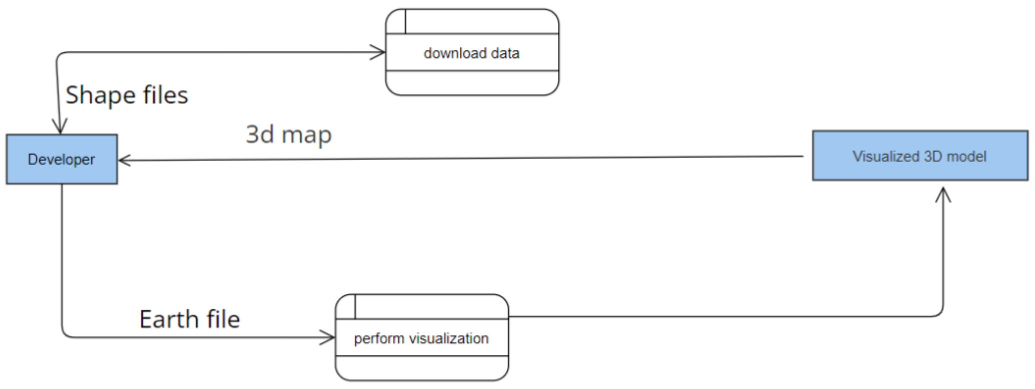


Fig 4.5 Data-flow diagram for Visualization

The whole project till now is implemented in five modules as shown below in fig (4.6). They are collection of datasets, PCA based image fusion, Wavelet based image fusion, statistical analysis on the fusion techniques and visualisation. The outcomes of these five Modules are also shown in respective segments.

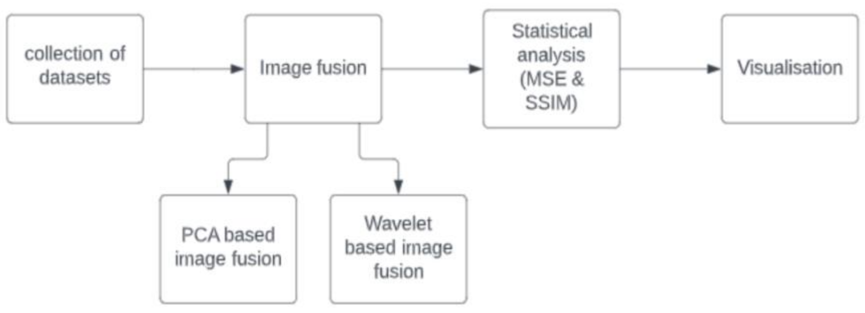


Fig 4.6 Flow of Modules

4.2 Module 1: Collection of data set

A dataset is a collection of related sets of information that is composed of separate elements but can be manipulated as a unit by a computer. These images are taken from sentinel-2 satellite. As mentioned earlier, there are 13 bands in a sentinel-2 satellite. So, the images are taken from random bands. These captured images are of the same size and of the same region but at different times. Image fusion can only be performed on the images of the same size. Data set is acquired from Open Access hub website. Images are of a region around Telangana-Karnataka border. The foot print of the region is fig (4.7)

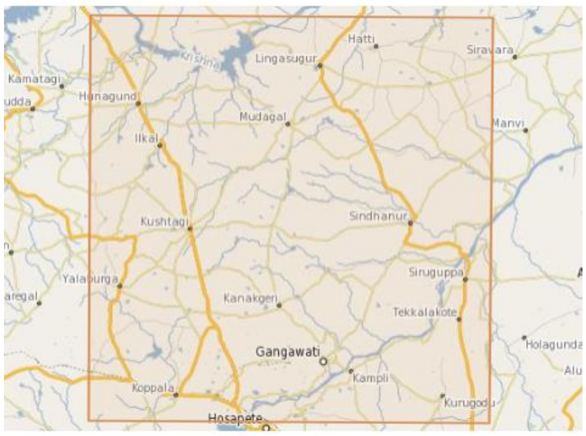


Fig 4.7 Foot print

4.3 Module 2: PCA based image fusion

Principal component analysis (PCA) is a statistical analysis for dimension reduction. It basically projects high dimensional data from a low dimensional space to increase the variance and reduce the covariance by retaining the components corresponding to the largest eigenvalues and discarding other components. It helps to reduce redundant information and highlight the components with biggest influence so as to increase the signal-to-noise ratio.

PCA is also a linear transformation that is easy to be implemented for applications in which huge amount of data is to be analyzed. It is widely used in data compression and pattern matching by expressing the data in a way to highlight the similarities and differences without much loss of information. PCA is employed to transform original image to its eigenspace. By retaining the principal components with influencing eigenvalues, it keeps the key features in the original image and reduces noise level.

We perform PCA on both the input images individually and the combine those images. As PCA retains the important features in an image, not much data is lost. The fusion is performed on those images and fused image will contain the best of two images. The architecture of PCA can be seen in fig (4.8).

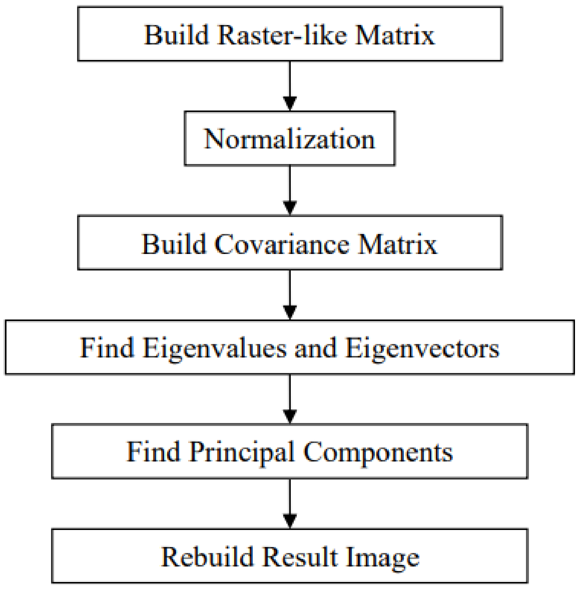


Fig 4.8 Architecture of PCA

4.4 Module 3: Wavelet based image fusion

The wavelets-based approach is appropriate for performing fusion tasks for the following reasons, it is a multiscale (multiresolution) approach well suited to manage the different image resolutions. In recent years, some researchers have studied multiscale representation (pyramid decomposition) of a signal and have established that multiscale information can be useful in a number of image processing applications including the image fusion.

The discrete wavelets transform (DWT) allows the image decomposition in different kinds of coefficients preserving the image information. Such coefficients coming from different images can be appropriately combined to obtain new coefficients, so that the information in the original images is collected appropriately. Once the coefficients are merged, the final fused image is achieved through the inverse DWT, where the information in the merged coefficients is also preserved. Fig (4.8) refers to architecture of wavelet-based image fusion.

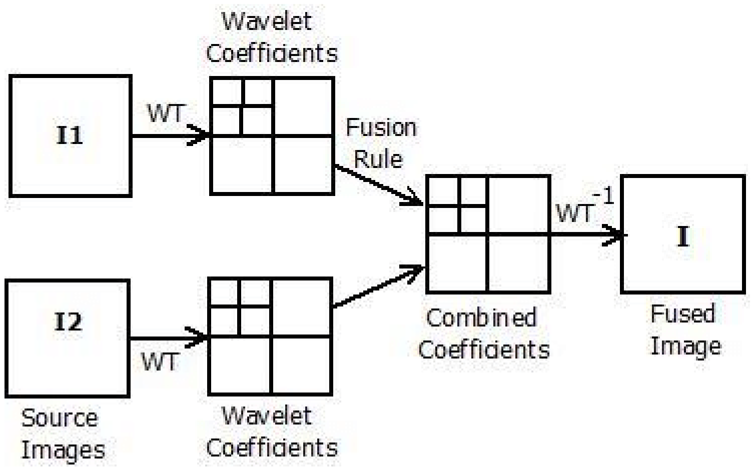


Fig 4.9 Architecture of Wavelet based Image fusion

4.5 Module 4: Statistical analysis

Mean Squared Error:

- MSE value denotes the average difference of the pixels all over the image.
- A higher value of MSE designates a greater difference amid the original image and processed image.
- The following equation provides a formula for calculation of the MSE.

$$\mathrm{MSE} = \frac{1}{N} \sum \sum (E_{ij} - o_{ij})^2$$

- Here, N is the image size, O is the original image, whereas, E is the edge image.

Structural Similarity Index:

- The Structural Similarity index is a method for measuring the similarity between two images or to measure the structural distortions of two images.
- The SSIM index can be viewed as a quality measure of one of the images being compared, provided the other image is regarded as of perfect quality.
- It attempts to model the perceived change in the structural information of the image. Also, the SSIM value can vary between -1 and 1, where 1 indicates perfect similarity.

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

4.6 Module 5: Visualization

Data visualization, simply put, is the graphical representation of data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. Virtual

Reality data visualizations are computer generated, highly interactive, 3D projects. The main goal of data visualization is to make it easier to identify patterns, trends and outliers in large data sets. osgEarth uses the familiar Map/Layer paradigm for organizing data. Refer to fig (4.9) to see the structure. The Map is comprised of a collection of layers. The renderer draws each visible layer one after the next, from bottom to top, to display the final scene. Enable the development of 3D geospatial appliations on top of OpenSceneGraph. Make it as easy as possible to visualize terrian models and 3D maps.

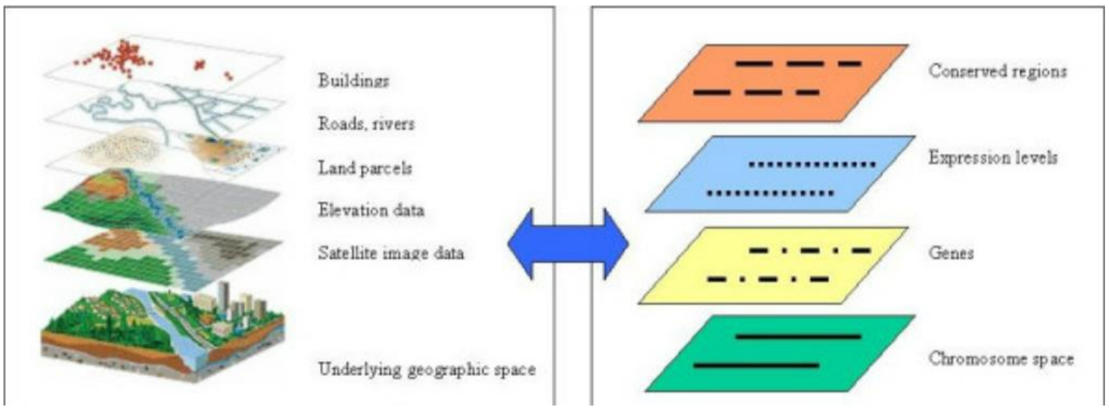


Fig 4.10 Map/Layer structure of OSGEarth

4.6.1 osgEarth software

OSGEarth is a geospatial SDK and terrain engine for OpenSceneGraph applications. The goals of osgEarth are to:

- Enable the development of 3D geospatial appliations on top of OpenSceneGraph.
- Make it as easy as possible to visualize terrian models and imagery directly from source data.
- Interoperate with open mapping standards, technologies, and data.

Since osgEarth is a free open source SDK, the source code is available to anyone and we welcome and encourage community participation when it comes to testing, adding features, and fixing bugs.

4.6.2 Installation

(OpenSceneGraph)

Download OpenSceneGraph as osgearth uses the libraries of OpenSceneGraph.

Steps to download OpenSceneGraph are:

1.1. Find MSYS2

1.2. Download MSYS2

1.3. Install MSYS2

1.4. Update MSYS2 core packages

1.5. Update MSYS2 system packages

1.6. Find MinGW-w64 CMake

1.7. Install build tools

1.8. Find OpenSceneGraph repository

1.9. Install git to get OpenSceneGraph

1.10. Get latest OpenSceneGraph

1.11. Create build director

1.12. Try to configure OpenSceneGraph with CMake

1.13. Add MinGW-w64 bin directory to PATH

1.14. Try to configure OpenSceneGraph once again

1.15. Observe configuration errors

1.16. Try to configure OpenSceneGraph for MinGW

1.17. Observe generator error

1.18. Clean build directory

1.19. Try to configure OpenSceneGraph for MinGW once again

1.20. Observe sh error

1.21. Update CMake MinGW script

1.22. Configure OpenSceneGraph

1.23. Build OpenSceneGraph

1.24. Try to install OpenSceneGraph

1.25. Observe installation error

1.26. Install OpenSceneGraph

1.27. Try to run 'osgviewer'

1.28. Observe missing library

1.29. Find MinGW libraries

1.30. Copy MinGW libraries

1.31. Go to the directory with 'osgviewer'

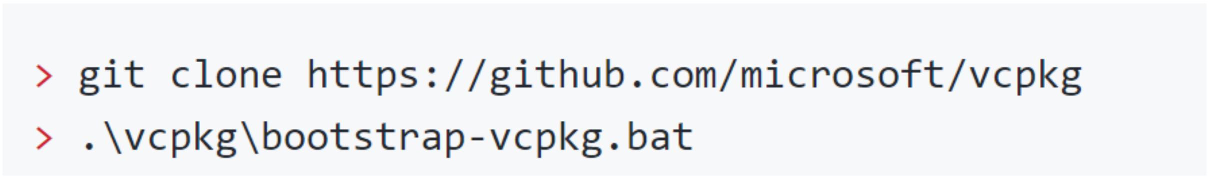
1.32. Check 'box.osgt' with 'osgviewer'

(osgEarth)

Step 1: Configure vcpkg

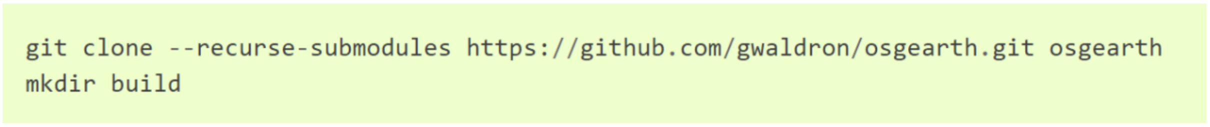
- First, download and bootstrap vcpkg:

  (latest version of git to be downloaded to clone)



```
> git clone https://github.com/microsoft/vcpkg
> .\vcpkg\bootstrap-vcpkg.bat
```

Step 2: Clone the repository



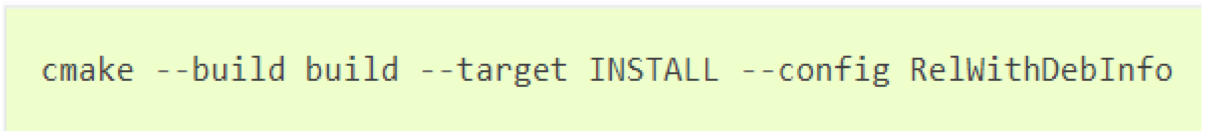
```
git clone --recurse-submodules https://github.com/gwaldron/osgearth.git osgearth
mkdir build
```

Step 3: Configure cmake

```
cmake -S osgearth -B build -G "Visual Studio 15 2017 Win64" -DCMAKE_BUILD_TYPE=RelWithDebInfo -DWIN32_USE_MP=ON -DCMAKE_INSTALL_PREFIX=[installroot] -DCMAKE_TOOLCHAIN_FILE=[vcpkgroot]\scripts\buildsystems\vcpkg.cmake
```

- In place of [vcpkgroot] in the command above, provide your path where you have cloned the repository.
- Please use Visual Studio 2015 or higher versions.
- Please make sure you install all files related to c++ in visual studio.

Step 4: Build and install osgEarth



```
cmake --build build --target INSTALL --config RelWithDebInfo
```

Step 5: Set up your runtime environment

- Make sure that the vcpkg dependencies and osgEarth are in your path.

4.6.3 osgEarth Command line tools

- **Osgearth_viewer:** Can load and display a map from and command line. The osgEarth EarthManipulator is used to control the camera and is optimized for viewing geospatial data.
- **Usage:**

osgearth_viewer filename.earth

- **Osgearth_version**: Tells us about the current version of the software.
- **Osgearth_toc**: Can load and display a map from the command line. The osgEarth EarthManipulator is used to control the camera and is optimized for viewing geospatial data. It also displays a TOC.

  Usage:

  osgearth_toc filename.earth

4.6.4 Layers

- **Image Layer:** An Image Layer is a raster image overlaid on the map's geometry.
  - ➢ GDAL: The GDAL plugin will read most geospatial file types. This is the most common driver that you will use to read data on your local filesystem.
  - ➢ XYZ: The XYZ plugin is useful for reading web map tile repositories with a standard X/Y/LOD setup but that don't explicitly report any metadata. Many of the popular web mapping services (like Map Quest) fall into this category. You need to provide osgEarth with a profile when using this driver.
  - ➢ TFS: This plugin reads vector data from a Tiled Feature Service repository. TFS is a tiled layout similar to TMS (Tile Map Service) but for cropped feature data.
  - ➢ OGR: This plugin reads vector data from any of the formats supported by the OGR Simple Feature Library (which is quite a lot). Most common among these includes ESRI Shapefiles, GML, and PostGIS.
- **Elevation Layer:** An Elevation Layer provides heightmap grids to the terrain engine. The osgEarth engine will composite all elevation data into a single heightmap and use that to build a terrain tile.
- **Feature geometry:(styling buildings, roads, rivers etc):**
  - ➢ Altitude: The altitude symbol (SDK: AltitudeSymbol) controls a feature's interaction with the terrain under its location.
  - ➢ Extrusion: The extrusion symbol (SDK: ExtrusionSymbol) directs osgEarth to create extruded geometry from the source vector data;

Extrusion turns a 2D vector into a 3D shape. Note: The simple presence of an extrusion property will enable extrusion.

- Skin: The skin symbol (SDK: SkinSymbol) applies texture mapping to a geometry, when applicable. (At the moment this only applies to extruded geometry.)
- Model: The model symbol (SDK: ModelSymbol) describes external 3D models.

5. RESULTS AND DISCUSSIONS

5.1 IMAGE FUSION

The results are obtained by performing the above, mentioned methods. We have applied PCA and the wavelet-based image fusion technique on five pairs of images from the data set we have collected. Table (5.1) contains the details of the image pairs.

On each pair we perform PCA and wavelet-based image fusions. After we get the output image which is a fused image, we perform statistical analysis on the input and output images. We take input image-1 of one pair and output image of the same pair and compute SSIM and MSE values. Similarly, we calculate SSIM and MSE values for input image-2 and output image of the pair. The same procedure is performed for all pairs. All these values are recorded to analyse the image fusion techniques.

Pair	Input	Resolution	Band
1	Image1	10m	B02 (Blue)
	Image2	20m	B02 (Blue)
2	Image1	20m	TCI
	Image2	60m	TCI
3	Image1	10m	TCI
	Image2	60m	TCI
4	Image1	10m	BO4(Red)
	Image2	20m	B04(Red)
5	Image1	10m	TCI
	Image2	20m	TCI

Table 5.1 Input images

The results of image fusion on the pairs of images in table (5.1) are mentioned in table (5.2), table (5.3), table (5.4), table (5.5) and table (5.6). And from fig (5.1) to fig (5.14) are the results of the statistical analysis performed on the input and output images.

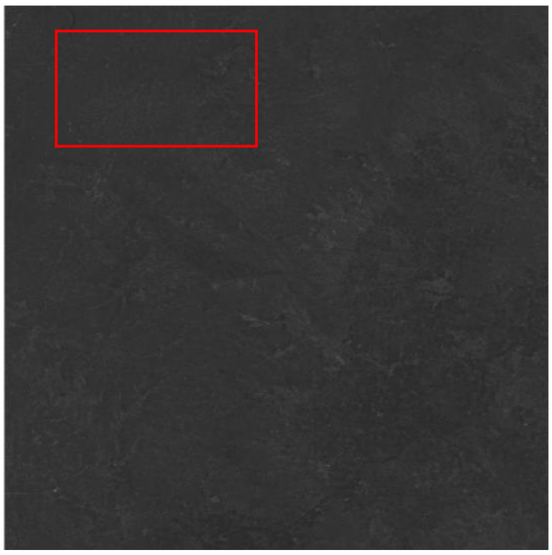
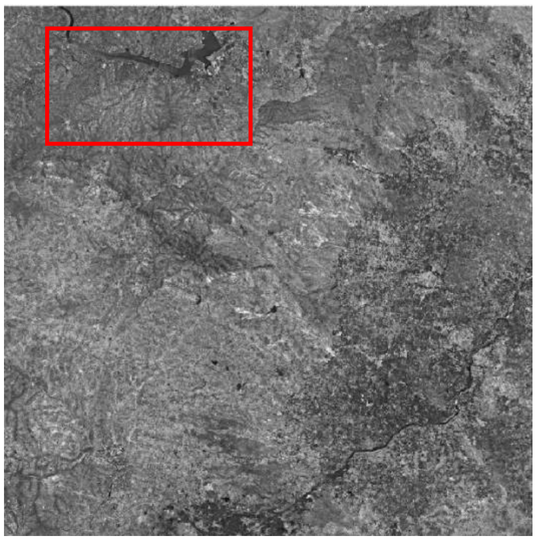
Pair no.	Input images	Fused output images
1		PCA based fused image  Wavelet based fused image 

Table 5.2 Image fusion results on Pair 1

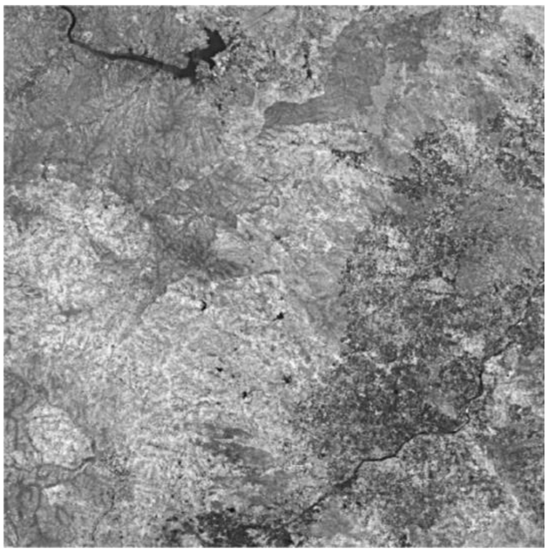
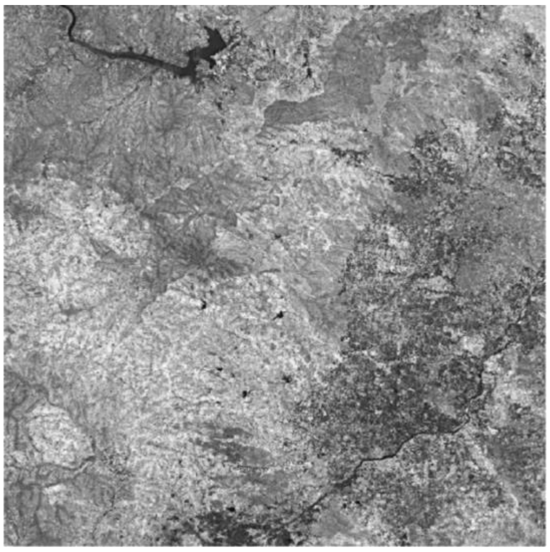
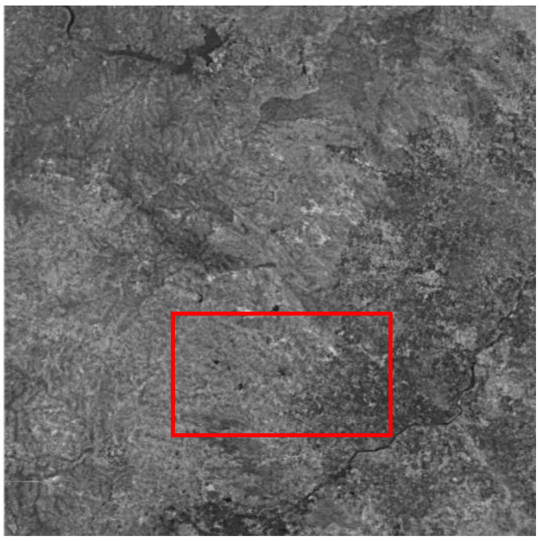
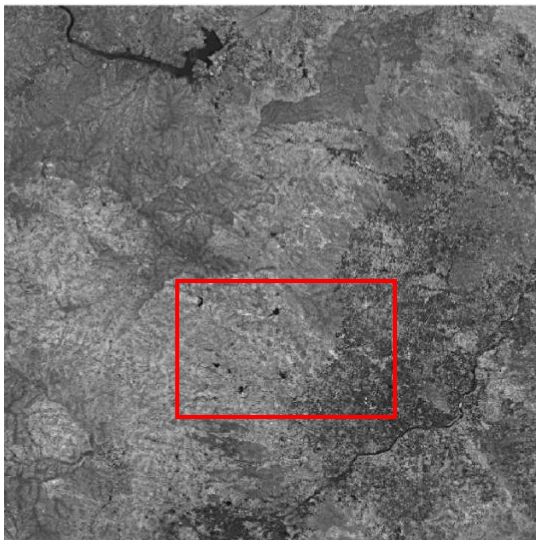
Pair no.	**Input images**	**Fused output images**
2	 	PCA based fused image 
		Wavelet based fused image 

Table 5.3 Image fusion results on Pair 2

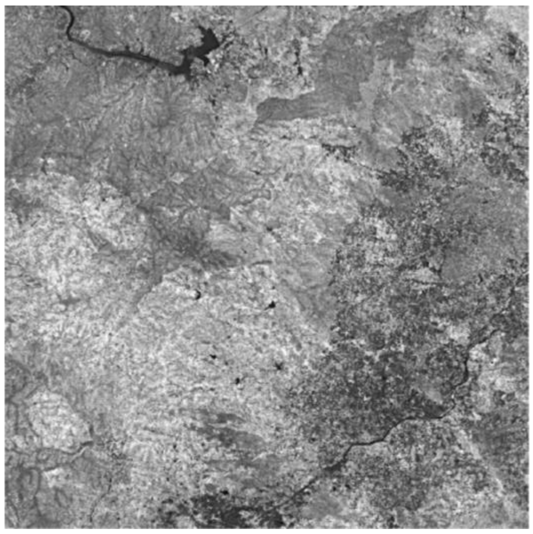
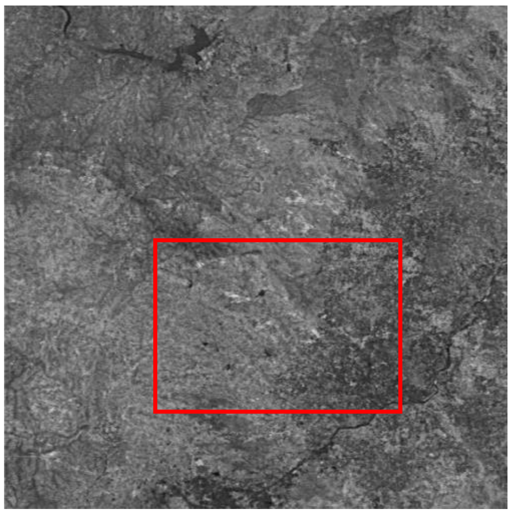
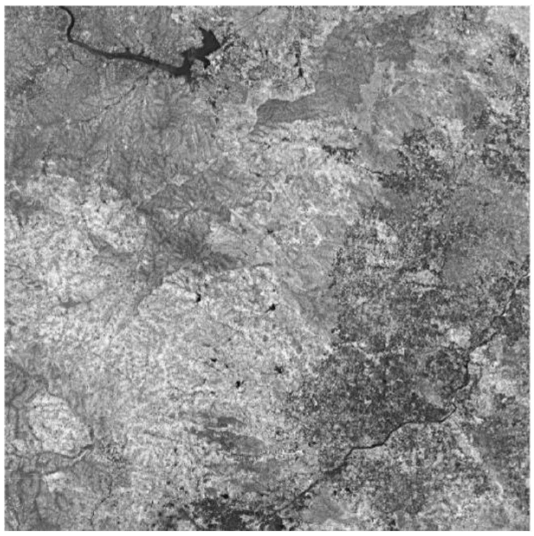
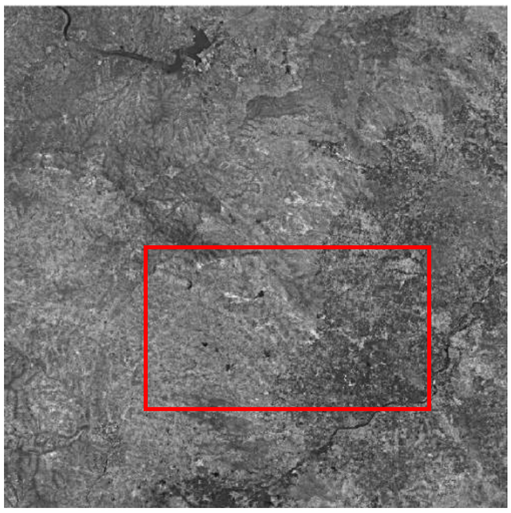
Pair no.	**Input images**	**Fused output images**
3		PCA based fused image 
		Wavelet based fused image 

Table 5.4 Image fusion results on Pair 3

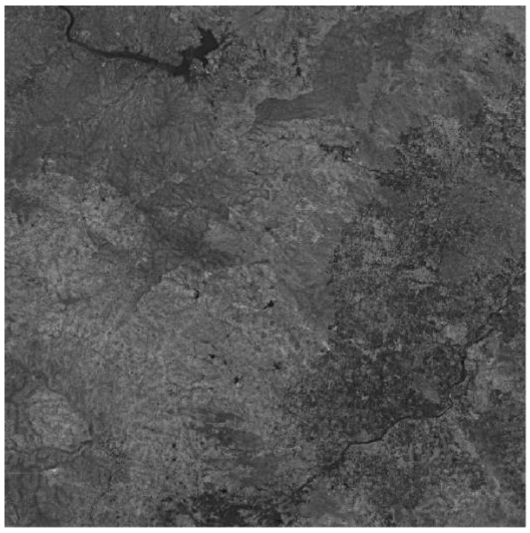
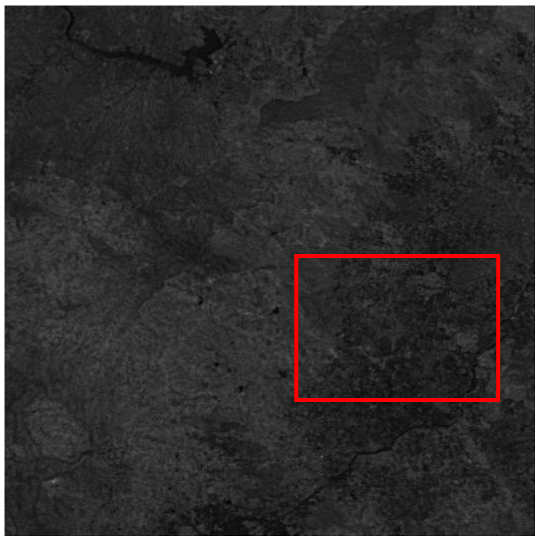
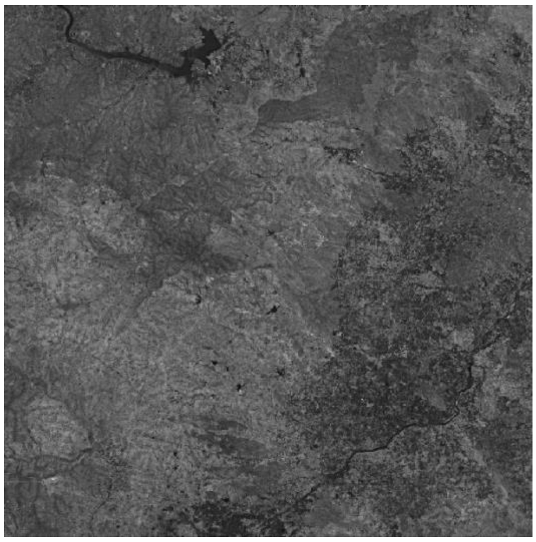
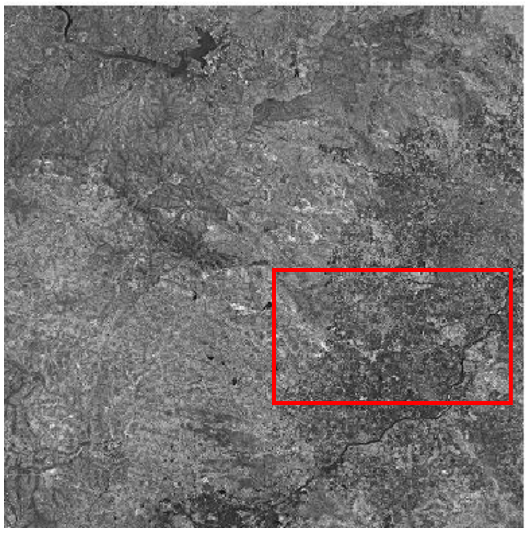
Pair no.	**Input images**	**Fused output images**
4		PCA based fused image 
		Wavelet based fused image 

Table 5.5 Image fusion results on Pair 4

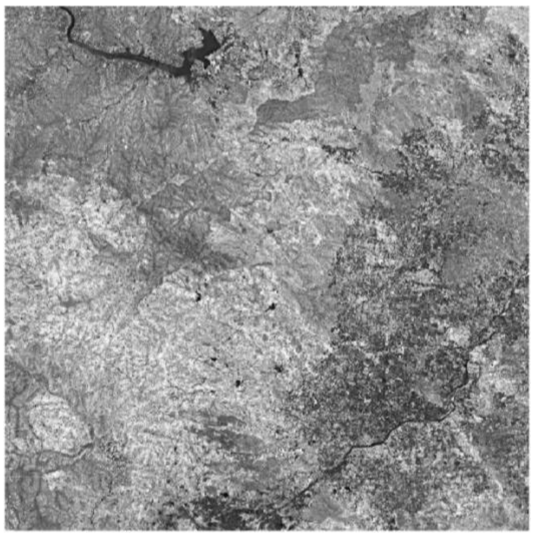
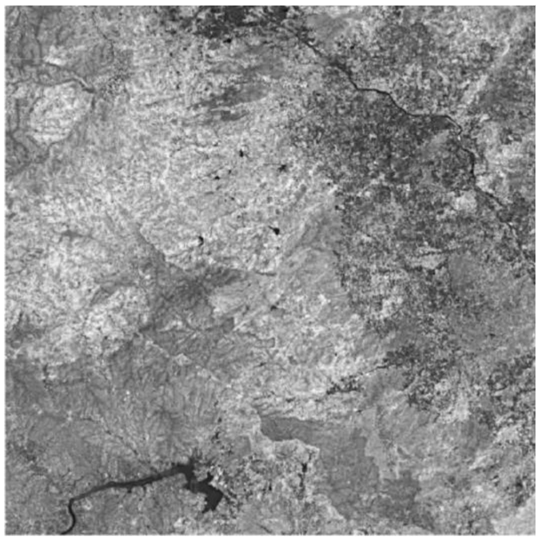
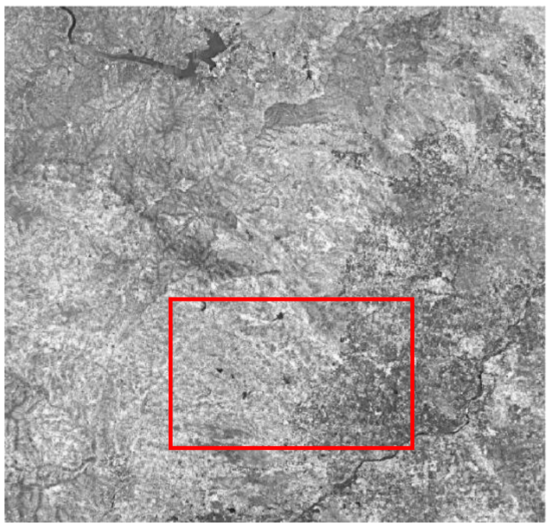
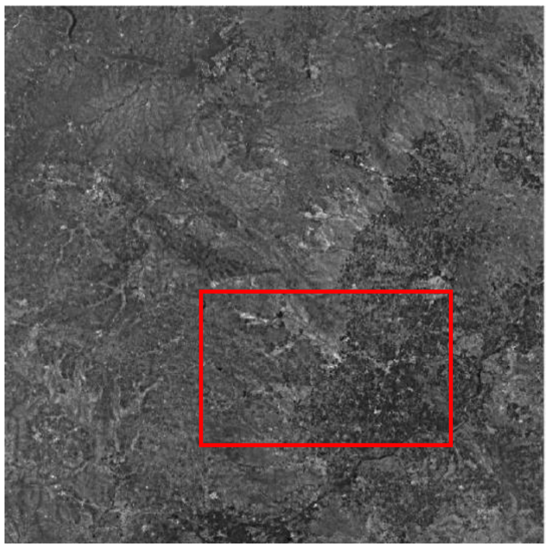
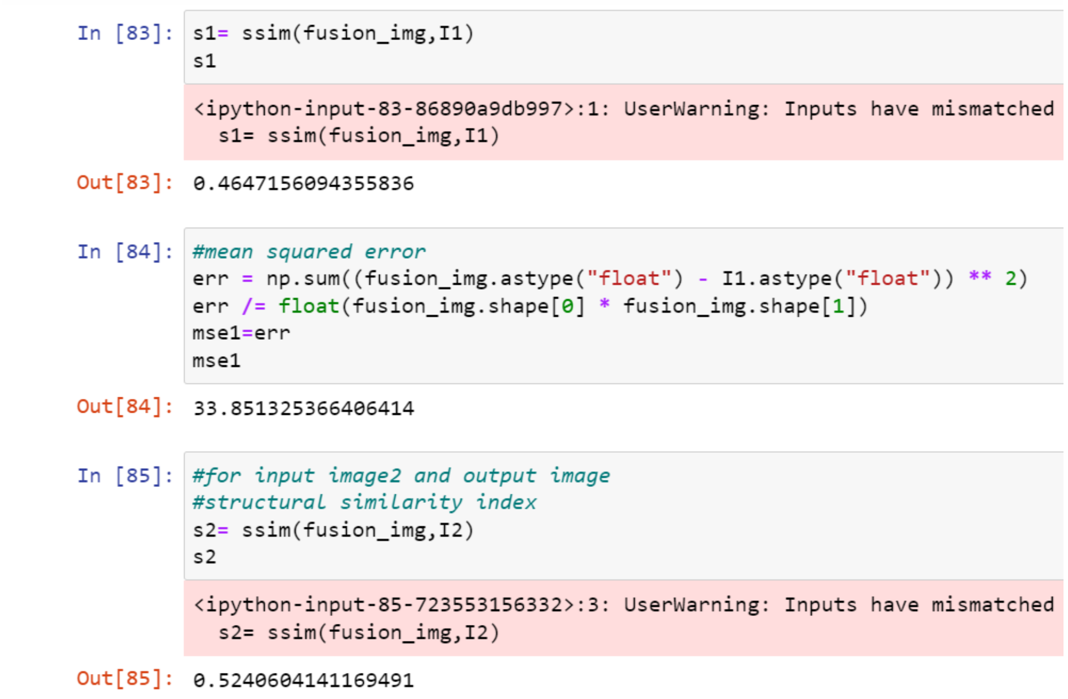
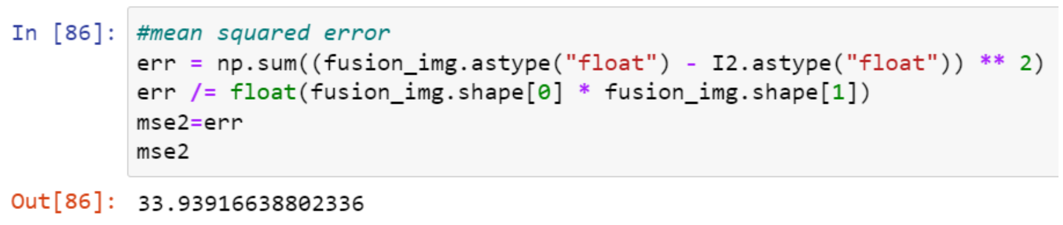
Pair no.	**Input images**	**Fused output images**
5	 	PCA based fused image 
		Wavelet based fused image 

Table 5.6 Image fusion results on Pair 5



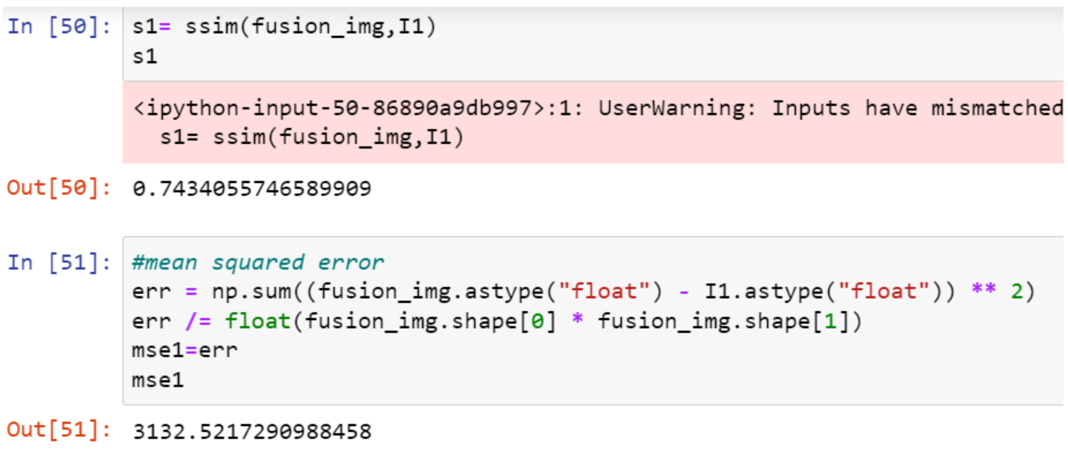
```
In [83]: s1= ssim(fusion_img,I1)
         s1

         <ipython-input-83-86890a9db997>:1: UserWarning: Inputs have mismatched
           s1= ssim(fusion_img,I1)

Out[83]: 0.4647156094355836

In [84]: #mean squared error
         err = np.sum((fusion_img.astype("float") - I1.astype("float")) ** 2)
         err /= float(fusion_img.shape[0] * fusion_img.shape[1])
         mse1=err
         mse1

Out[84]: 33.851325366406414

In [85]: #for input image2 and output image
         #structural similarity index
         s2= ssim(fusion_img,I2)
         s2

         <ipython-input-85-723553156332>:3: UserWarning: Inputs have mismatched
           s2= ssim(fusion_img,I2)

Out[85]: 0.5240604141169491
```

Fig 5.1 statistical analysis results on pair-1 (PCA based image fusion)



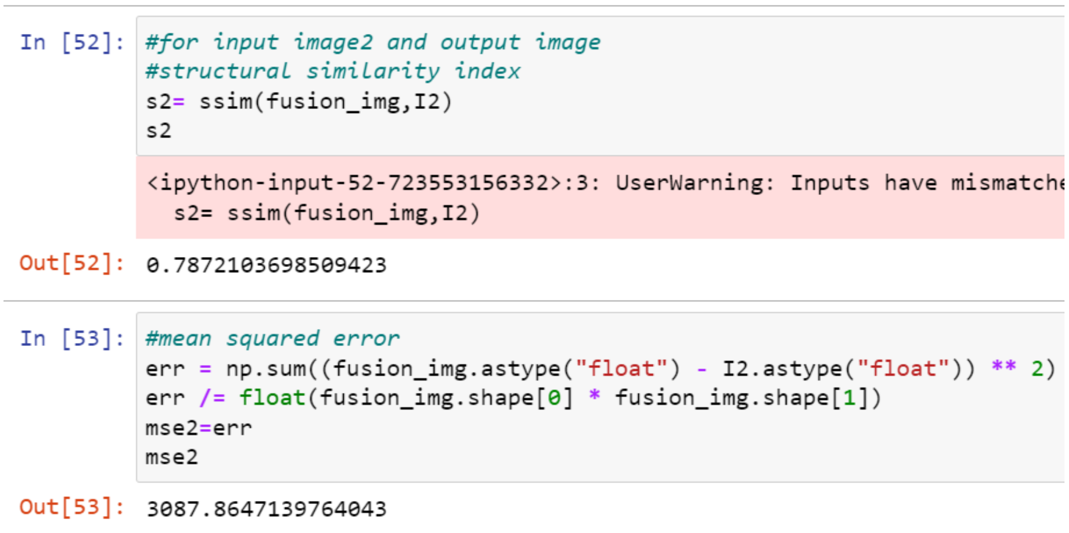
```
In [86]: #mean squared error
         err = np.sum((fusion_img.astype("float") - I2.astype("float")) ** 2)
         err /= float(fusion_img.shape[0] * fusion_img.shape[1])
         mse2=err
         mse2

Out[86]: 33.93916638802336
```

Fig 5.2 statistical analysis results on pair-1 (PCA based image fusion)



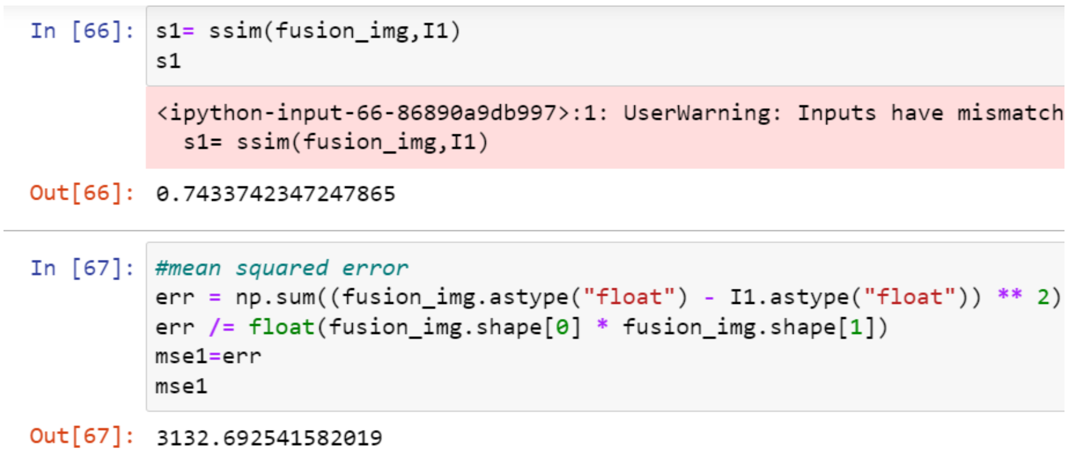
```
In [50]: s1= ssim(fusion_img,I1)
         s1

         <ipython-input-50-86890a9db997>:1: UserWarning: Inputs have mismatched
           s1= ssim(fusion_img,I1)

Out[50]: 0.7434055746589909

In [51]: #mean squared error
         err = np.sum((fusion_img.astype("float") - I1.astype("float")) ** 2)
         err /= float(fusion_img.shape[0] * fusion_img.shape[1])
         mse1=err
         mse1

Out[51]: 3132.5217290988458
```

Fig 5.3 statistical analysis results on pair-2 (PCA based image fusion)



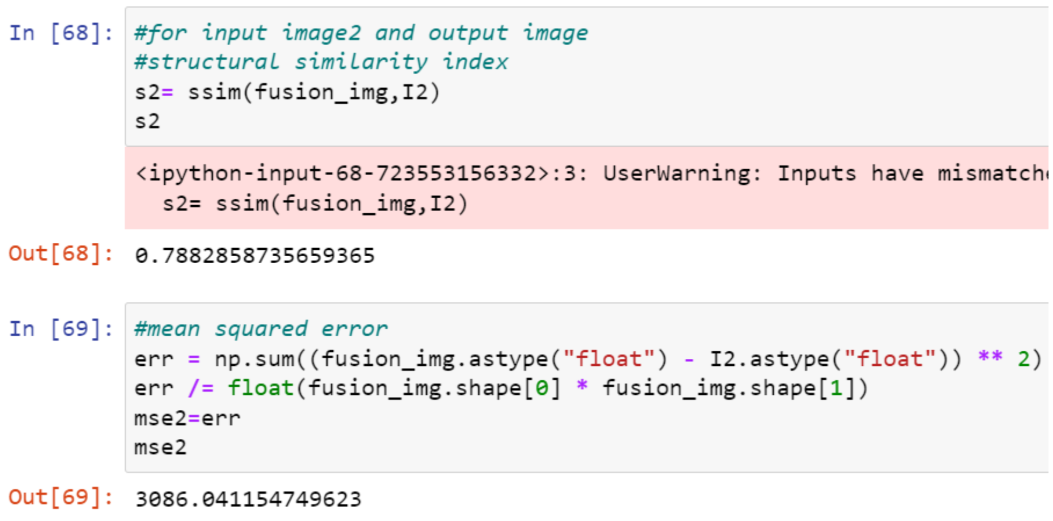
```
In [52]: #for input image2 and output image
         #structural similarity index
         s2= ssim(fusion_img,I2)
         s2

         <ipython-input-52-723553156332>:3: UserWarning: Inputs have mismatche
           s2= ssim(fusion_img,I2)

Out[52]: 0.7872103698509423

In [53]: #mean squared error
         err = np.sum((fusion_img.astype("float") - I2.astype("float")) ** 2)
         err /= float(fusion_img.shape[0] * fusion_img.shape[1])
         mse2=err
         mse2

Out[53]: 3087.8647139764043
```

Fig 5.4 statistical analysis results on pair-2 (PCA based image fusion)



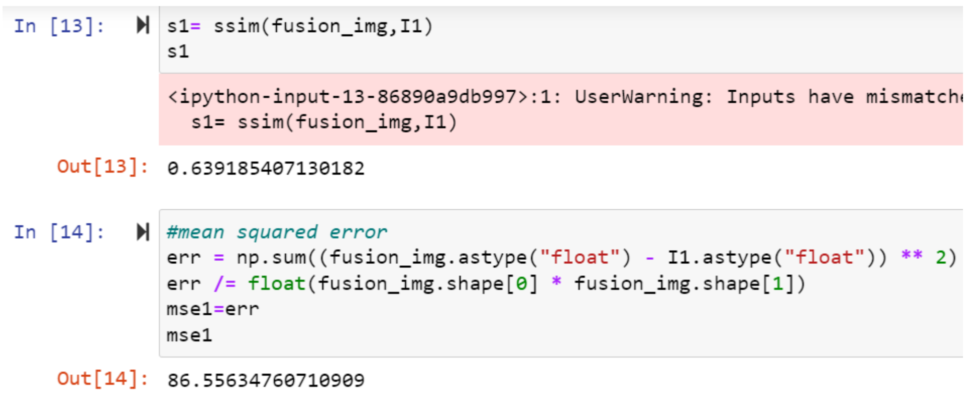
```
In [66]: s1= ssim(fusion_img,I1)
         s1

         <ipython-input-66-86890a9db997>:1: UserWarning: Inputs have mismatch
           s1= ssim(fusion_img,I1)

Out[66]: 0.7433742347247865

In [67]: #mean squared error
         err = np.sum((fusion_img.astype("float") - I1.astype("float")) ** 2)
         err /= float(fusion_img.shape[0] * fusion_img.shape[1])
         mse1=err
         mse1

Out[67]: 3132.692541582019
```

Fig 5.5 statistical analysis results on pair-3 (PCA based image fusion)



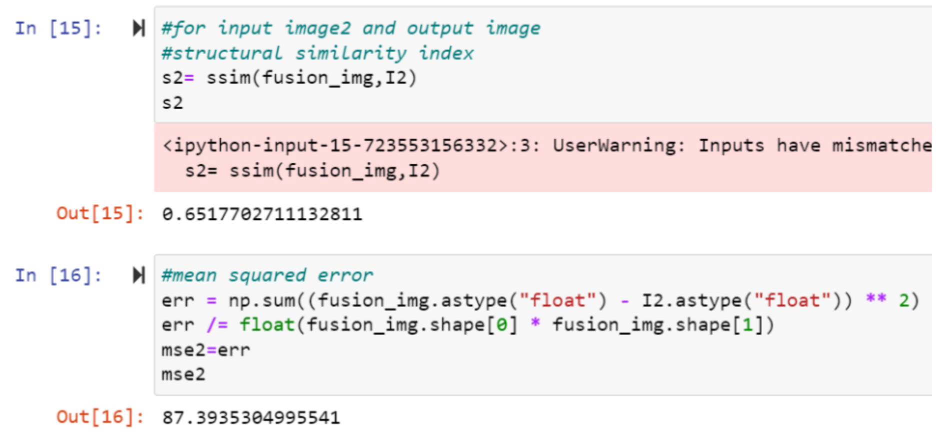
```
In [68]: #for input image2 and output image
         #structural similarity index
         s2= ssim(fusion_img,I2)
         s2

         <ipython-input-68-723553156332>:3: UserWarning: Inputs have mismatche
           s2= ssim(fusion_img,I2)

Out[68]: 0.7882858735659365

In [69]: #mean squared error
         err = np.sum((fusion_img.astype("float") - I2.astype("float")) ** 2)
         err /= float(fusion_img.shape[0] * fusion_img.shape[1])
         mse2=err
         mse2

Out[69]: 3086.041154749623
```

Fig 5.6 statistical analysis results on pair-3 (PCA based image fusion)



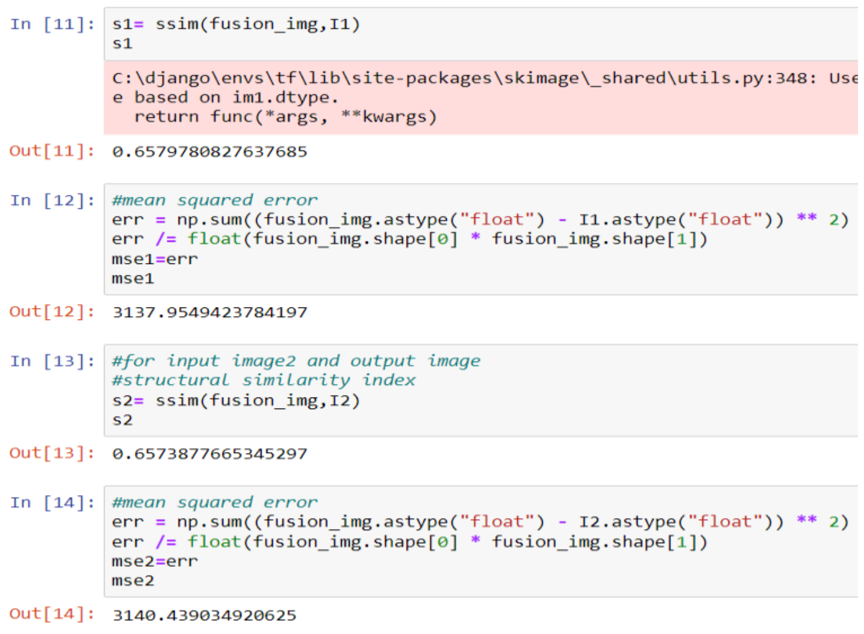
```
In [13]:  s1= ssim(fusion_img,I1)
          s1

          <ipython-input-13-86890a9db997>:1: UserWarning: Inputs have mismatche
            s1= ssim(fusion_img,I1)

Out[13]:  0.639185407130182

In [14]:  #mean squared error
          err = np.sum((fusion_img.astype("float") - I1.astype("float")) ** 2)
          err /= float(fusion_img.shape[0] * fusion_img.shape[1])
          mse1=err
          mse1

Out[14]:  86.55634760710909
```

Fig 5.7 statistical analysis results on pair-4 (PCA based image fusion)



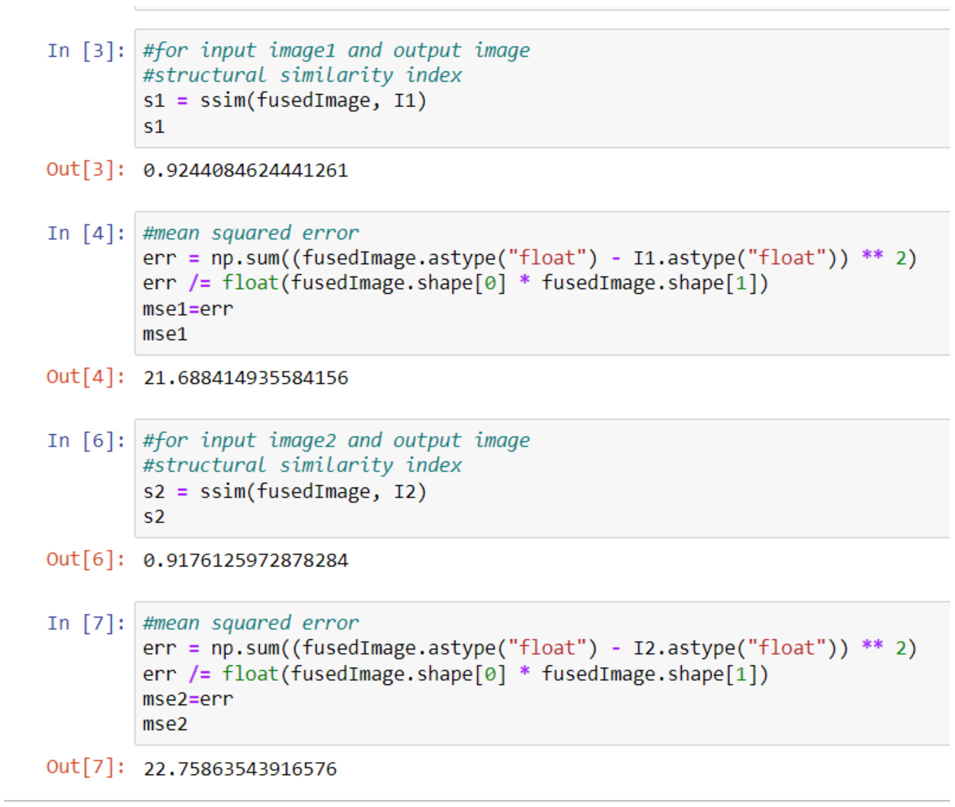
```
In [15]:  #for input image2 and output image
          #structural similarity index
          s2= ssim(fusion_img,I2)
          s2

          <ipython-input-15-723553156332>:3: UserWarning: Inputs have mismatche
            s2= ssim(fusion_img,I2)

Out[15]:  0.6517702711132811

In [16]:  #mean squared error
          err = np.sum((fusion_img.astype("float") - I2.astype("float")) ** 2)
          err /= float(fusion_img.shape[0] * fusion_img.shape[1])
          mse2=err
          mse2

Out[16]:  87.3935304995541
```

Fig 5.8 statistical analysis results on pair-4 (PCA based image fusion)



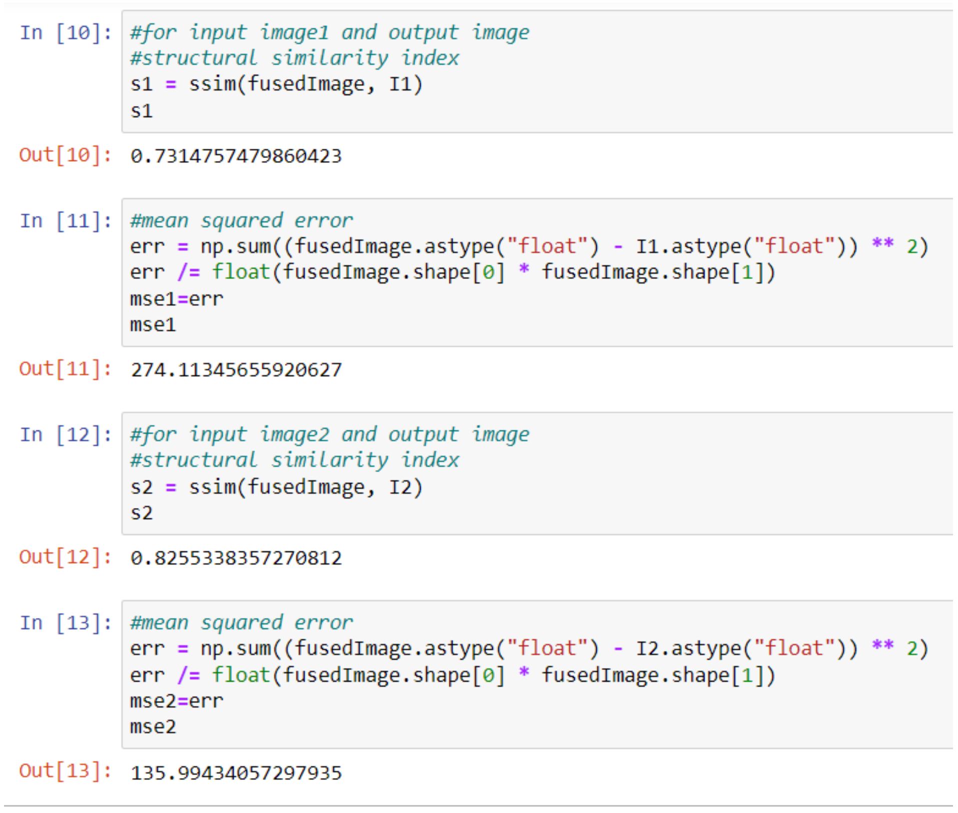
```
In [11]:  s1= ssim(fusion_img,I1)
          s1

          C:\django\envs\tf\lib\site-packages\skimage\_shared\utils.py:348: Use
          e based on im1.dtype.
            return func(*args, **kwargs)

Out[11]:  0.6579780827637685

In [12]:  #mean squared error
          err = np.sum((fusion_img.astype("float") - I1.astype("float")) ** 2)
          err /= float(fusion_img.shape[0] * fusion_img.shape[1])
          mse1=err
          mse1

Out[12]:  3137.9549423784197

In [13]:  #for input image2 and output image
          #structural similarity index
          s2= ssim(fusion_img,I2)
          s2

Out[13]:  0.6573877665345297

In [14]:  #mean squared error
          err = np.sum((fusion_img.astype("float") - I2.astype("float")) ** 2)
          err /= float(fusion_img.shape[0] * fusion_img.shape[1])
          mse2=err
          mse2

Out[14]:  3140.439034920625
```

Fig 5.9 statistical analysis results on pair-5 (PCA based image fusion)



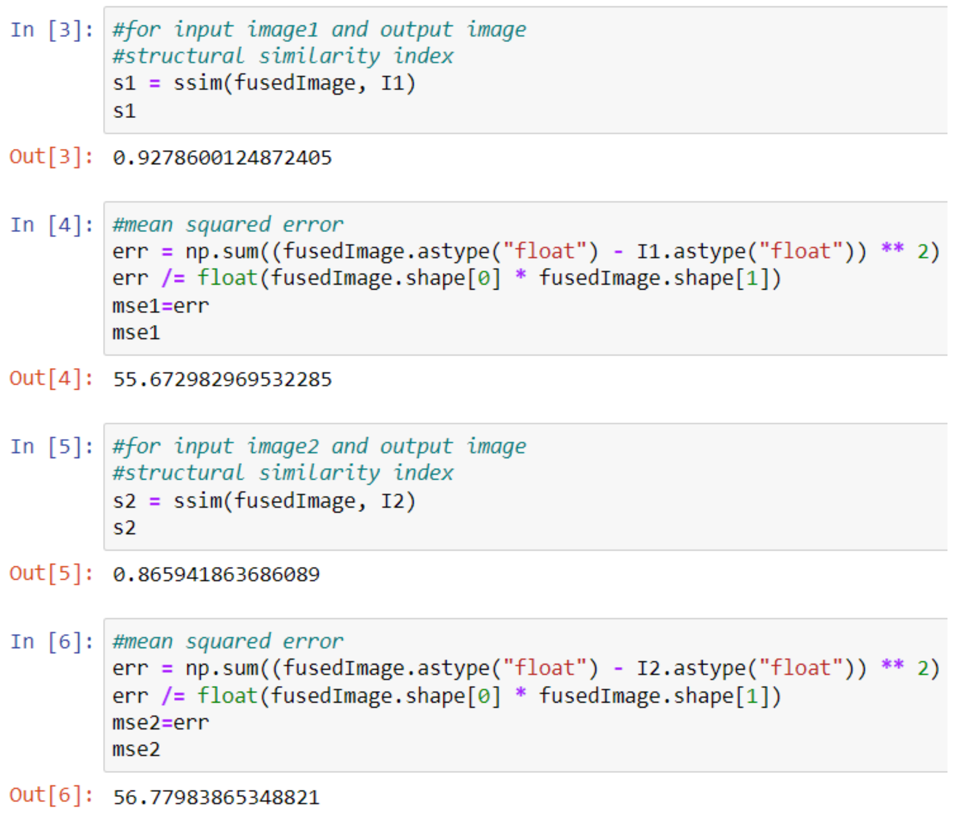
```
In [3]: #for input image1 and output image
        #structural similarity index
        s1 = ssim(fusedImage, I1)
        s1

Out[3]: 0.9244084624441261

In [4]: #mean squared error
        err = np.sum((fusedImage.astype("float") - I1.astype("float")) ** 2)
        err /= float(fusedImage.shape[0] * fusedImage.shape[1])
        mse1=err
        mse1

Out[4]: 21.688414935584156

In [6]: #for input image2 and output image
        #structural similarity index
        s2 = ssim(fusedImage, I2)
        s2

Out[6]: 0.9176125972878284

In [7]: #mean squared error
        err = np.sum((fusedImage.astype("float") - I2.astype("float")) ** 2)
        err /= float(fusedImage.shape[0] * fusedImage.shape[1])
        mse2=err
        mse2

Out[7]: 22.75863543916576
```

Fig 5.10 statistical analysis results on pair-1 (Wavelet-based image fusion)



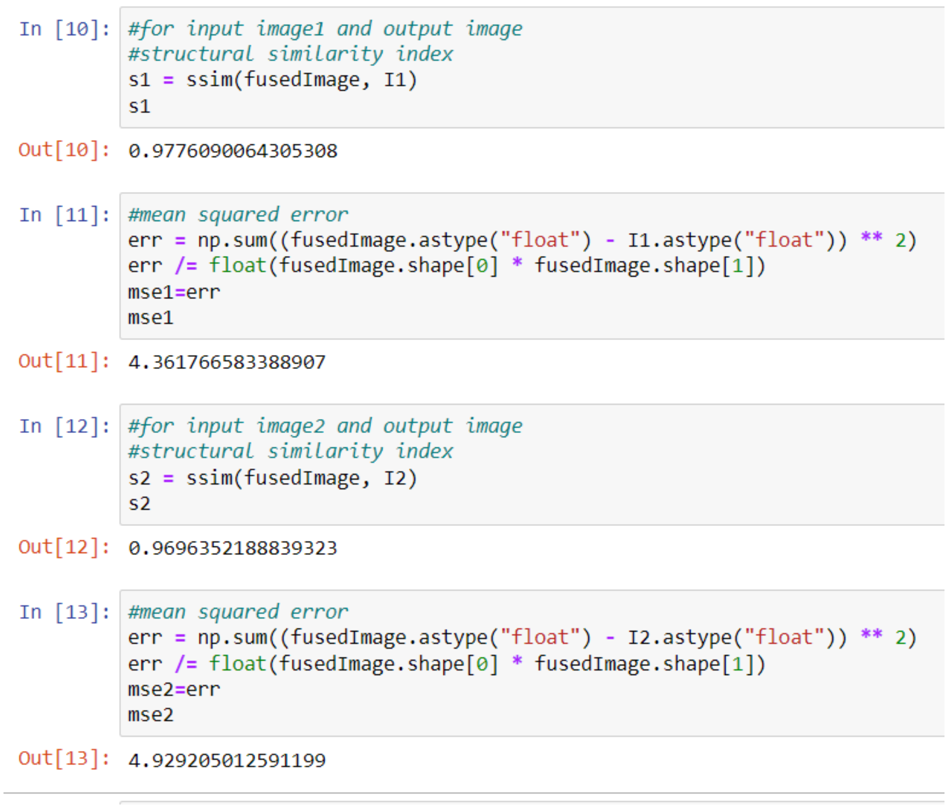
```
In [10]: #for input image1 and output image
         #structural similarity index
         s1 = ssim(fusedImage, I1)
         s1

Out[10]: 0.7314757479860423

In [11]: #mean squared error
         err = np.sum((fusedImage.astype("float") - I1.astype("float")) ** 2)
         err /= float(fusedImage.shape[0] * fusedImage.shape[1])
         mse1=err
         mse1

Out[11]: 274.11345655920627

In [12]: #for input image2 and output image
         #structural similarity index
         s2 = ssim(fusedImage, I2)
         s2

Out[12]: 0.8255338357270812

In [13]: #mean squared error
         err = np.sum((fusedImage.astype("float") - I2.astype("float")) ** 2)
         err /= float(fusedImage.shape[0] * fusedImage.shape[1])
         mse2=err
         mse2

Out[13]: 135.99434057297935
```

Fig 5.11 statistical analysis results on pair-2 (Wavelet-based image fusion)



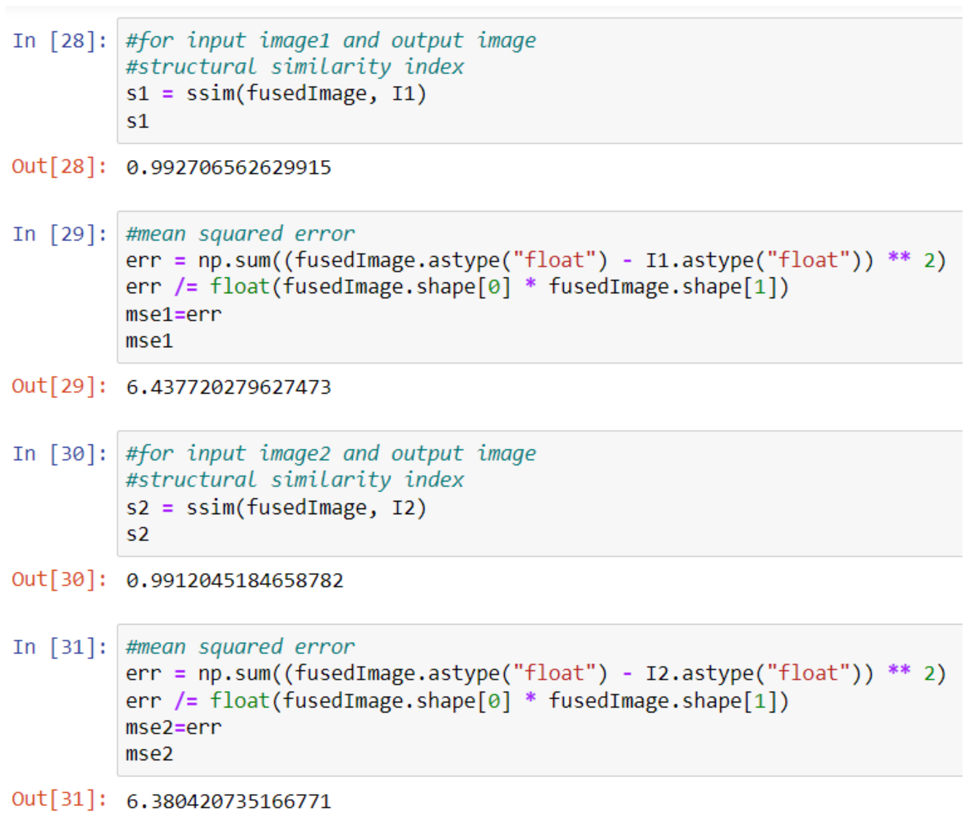
```
In [3]: #for input image1 and output image
        #structural similarity index
        s1 = ssim(fusedImage, I1)
        s1

Out[3]: 0.9278600124872405

In [4]: #mean squared error
        err = np.sum((fusedImage.astype("float") - I1.astype("float")) ** 2)
        err /= float(fusedImage.shape[0] * fusedImage.shape[1])
        mse1=err
        mse1

Out[4]: 55.672982969532285

In [5]: #for input image2 and output image
        #structural similarity index
        s2 = ssim(fusedImage, I2)
        s2

Out[5]: 0.865941863686089

In [6]: #mean squared error
        err = np.sum((fusedImage.astype("float") - I2.astype("float")) ** 2)
        err /= float(fusedImage.shape[0] * fusedImage.shape[1])
        mse2=err
        mse2

Out[6]: 56.77983865348821
```

Fig 5.12 statistical analysis results on pair-3 (Wavelet-based image fusion)



```
In [10]: #for input image1 and output image
         #structural similarity index
         s1 = ssim(fusedImage, I1)
         s1

Out[10]: 0.9776090064305308

In [11]: #mean squared error
         err = np.sum((fusedImage.astype("float") - I1.astype("float")) ** 2)
         err /= float(fusedImage.shape[0] * fusedImage.shape[1])
         mse1=err
         mse1

Out[11]: 4.361766583388907

In [12]: #for input image2 and output image
         #structural similarity index
         s2 = ssim(fusedImage, I2)
         s2

Out[12]: 0.9696352188839323

In [13]: #mean squared error
         err = np.sum((fusedImage.astype("float") - I2.astype("float")) ** 2)
         err /= float(fusedImage.shape[0] * fusedImage.shape[1])
         mse2=err
         mse2

Out[13]: 4.929205012591199
```

Fig 5.13 statistical analysis results on pair-4 (Wavelet-based image fusion)



```
In [28]: #for input image1 and output image
         #structural similarity index
         s1 = ssim(fusedImage, I1)
         s1

Out[28]: 0.992706562629915

In [29]: #mean squared error
         err = np.sum((fusedImage.astype("float") - I1.astype("float")) ** 2)
         err /= float(fusedImage.shape[0] * fusedImage.shape[1])
         mse1=err
         mse1

Out[29]: 6.437720279627473

In [30]: #for input image2 and output image
         #structural similarity index
         s2 = ssim(fusedImage, I2)
         s2

Out[30]: 0.9912045184658782

In [31]: #mean squared error
         err = np.sum((fusedImage.astype("float") - I2.astype("float")) ** 2)
         err /= float(fusedImage.shape[0] * fusedImage.shape[1])
         mse2=err
         mse2

Out[31]: 6.380420735166771
```

Fig 5.14 statistical analysis results on pair-5 (Wavelet-based image fusion)

The results from the statistical analysis are put together in tables below. The results of statistical analysis on PCA based image fusion are mentioned in table (5.7) and the results of statistical analysis on wavelet-based image fusion are mentioned in table (5.8).

Pair	**Input x Fused image**	**PCA based Image fusion (x fusedimagepca)**	
		SSIM	**MSE**
1	I1	0.57032775006	33.60610287746343
	I2	0.6128895379214	33.870180745903205
2	I1	0.74340557465	3132.5217290988458
	I2	0.787210369850	3087.864739764
3	I1	0.743374234724	3132.692541582019
	I2	0.78828587356	3086.041154749623
4	I1	0.639185407130182	86.55634760710909

	I2	0.6517702711132811	87.3935304995541
5	I1	0.6579780827637685	3137.9549423784197
	I2	0.6573877665345297	3140.439034920625

Table 5.7 PCA Statistical analysis

Pair	**Input x Fused image**	**Wavelet based Image fusion (x fusedimagewavelet)**	
		SSIM	**MSE**
1	I1	0.9244084624441261	21.688414935584156
	I2	0.917612597287284	22.75863543916576
2	I1	0.731475749860432	274.11345655920627
	I2	0.8255338357270812	135.993434057297935
3	I1	0.9278600124872405	55.672982969532285
	I2	0.865941863686089	56.77983865348821
4	I1	0.9776090064305308	4.361766583388907
	I2	0.9696352188839323	4.929205012591199
5	I1	0.992706562629915	6.437720279627473
	I2	0.9912045184658782	6.380420735166771

Table 5.8 Wavelet Statistical analysis

It has been observed that the MSE values of wavelet-based image fusion techniques are lower than that of PCA based image fusion technique and the SSIM values of wavelet-based image fusion technique are higher than that of PCA based image fusion technique. By these observations, it is been proven that wavelet-based image fusion technique works better the PCA based image fusion technique.

5.2 VISUALIZATION

The figures (5.15, 5.16, 5.17) are the just mapbox satellite images of Delhi, Bangalore, Chennai cities. Additional layer has been added and with the help of css we were able to visualize buildings in that area, that can be seen in figures (5.18, 5.19). Fig (5.20) shows the districts in India separated by red color boundary line.

Fig 5.15 Satellite image of Delhi (mapbox satellite)

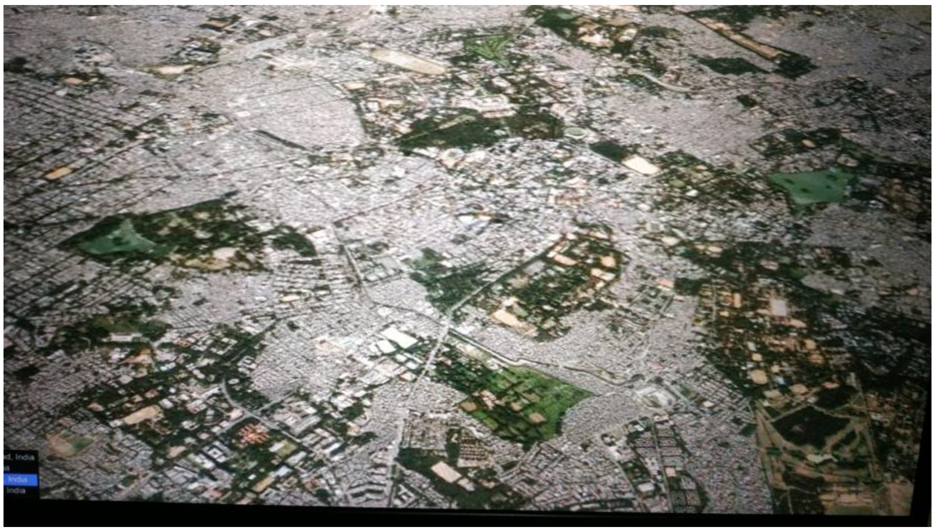


Fig 5.16 Satellite image of Bangalore (mapbox satellite)

Fig 5.17 Satellite image of Chennai (mapbox satellite)

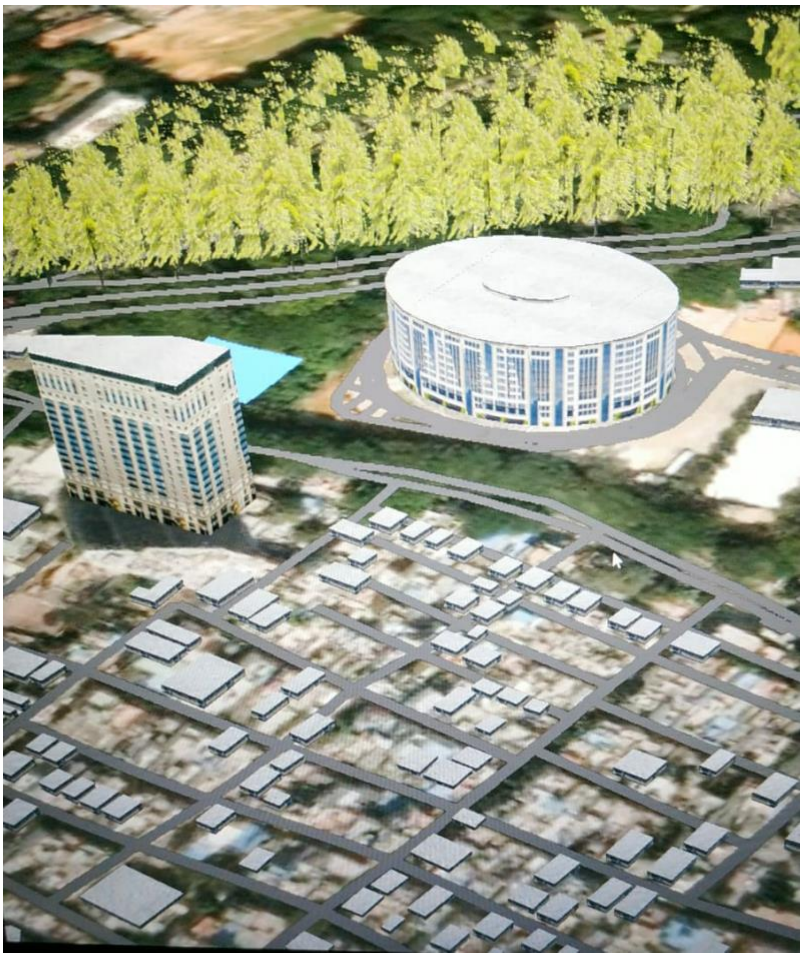


Fig 5.18 Buildings of Bangalore city

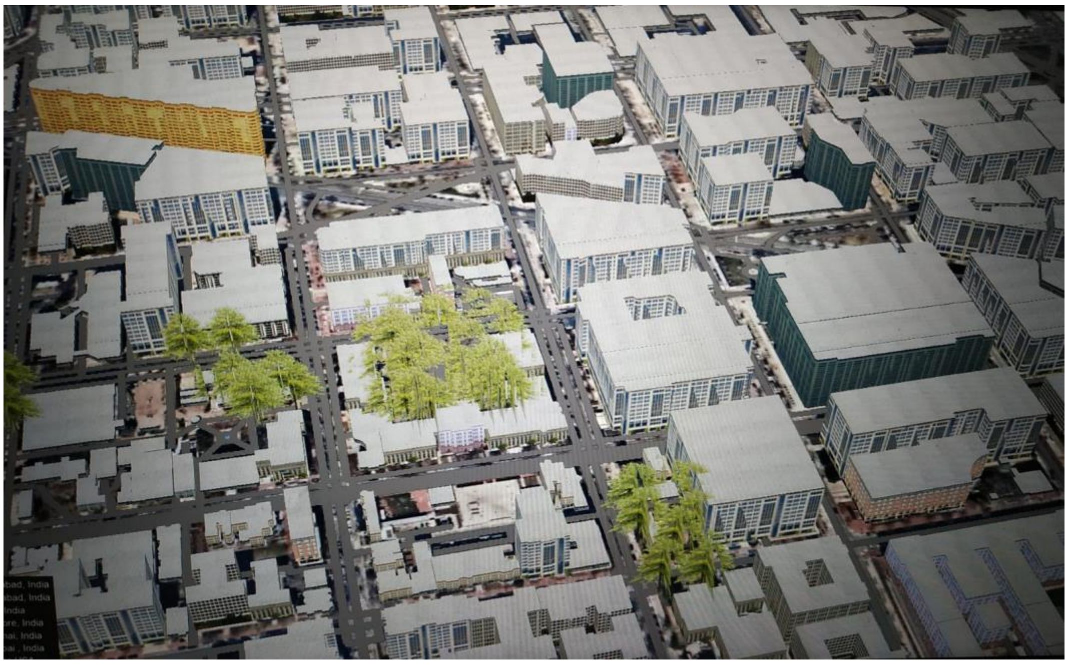


Fig 5.19 Mexico (USA) Buildings

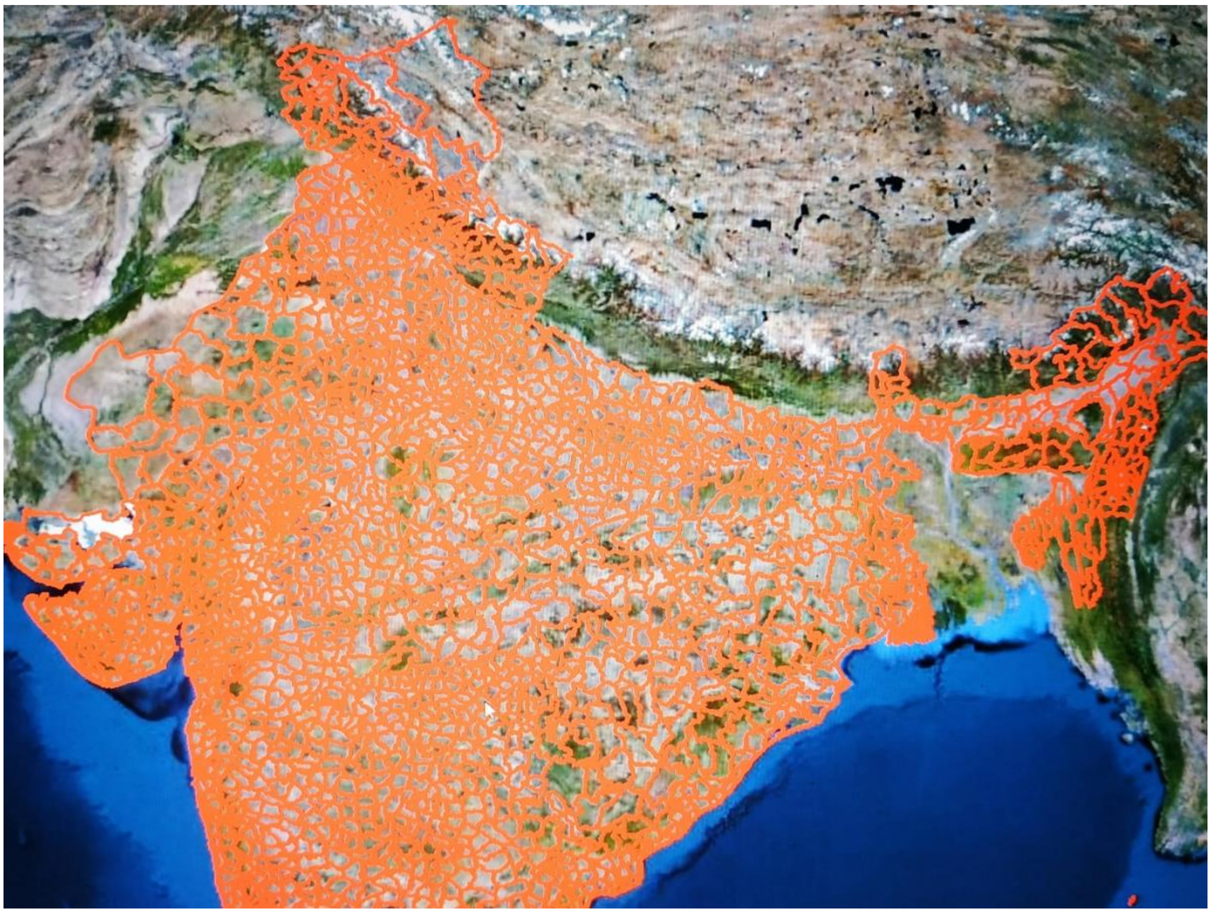


Fig 5.20 Indian districts (shape file)

6. CONCLUSION AND FUTURE SCOPE

We performed image fusion on sentinel data using machine learning algorithms like PCA and wavelet in our work. The results show that the algorithms working on the sentinel-2 satellite data. They give a unique output individually. To estimate which technique would be more suitable for sentinel-2 imagery, statistical analysis has been carried out on the images. After observing the values of MSE and SSIM, it is clear that wavelet-based image fusion technique is more suitable for sentinel-2 imagery.

In the future, we are planning to implement image fusion using deep learning as it has become a wide range of study for many researchers and might show better results than the other machine learning algorithms. The fused image can be visualized using OSG Earth to get better insight of the area/region. Further, instead of fusing sentinel images among themselves, they can be fused with other satellite images like IRS satellite images. This will increase the level of detailing of the region because of the spatial resolution of IRS images.

REFERENCES

[1] Scarpa, Giuseppe & Gargiulo, Massimiliano & Mazza, Antonio & Gaetano, Raffaele. (2018). *A CNN-Based Fusion Method for Feature Extraction from Sentinel Data. Remote Sensing*. 10. 236. 10.3390/rs10020236.

[2] Metwalli, Mohamed & Nasr, A.H. & Faragallah, Osama & El-Rabaie, El-Sayed. (2010). *Image Fusion Based on Principal Component Analysis and High-Pass Filter*. 63 - 70. 10.1109/ICCES.2009.5383308.

[3] Krista Amolins∗, Yun Zhang, Peter Dare, *Wavelet based image fusion techniques — An introduction, review and comparison*, ISPRS Journal of Photogrammetry & Remote Sensing 62 (2007) 249 – 263.

[4] Kaur, H., Koundal, D. & Kadyan, V. Image Fusion Techniques: A Survey. Arch Computat Methods Eng 28, 4425–4447 (2021).

[5] Size Li, Pengjiang Qian, Xin Zhang, Aiguo Chen, *Research on Image Denoising and Super-Resolution Reconstruction Technology of Multiscale-Fusion Images*, Mobile Information Systems, vol. 2021, Article ID 5184688, 11 pages, 2021.

[6] Andrea Garzelli, *A Review of Image Fusion Algorithms Based on the Super-Resolution Paradigm*, Remote Sens. 2016, *8*, 797.

[7] Qeethara Al-Shayea, Muzhir Al-Ani, *Efficient 3D Object Visualization via 2D Images*.

[8] Ganesan Gunasekaran and Meenakshisundaram Venkatesan, *An Efficient Technique for Three-Dimensional Image Visualization Through Two-Dimensional Images for Medical Data*.

[9] Steps to download OpenSceneGraph

APPENDIX

Code:

PCA based image fusion:

```
import os

import pywt

from os import environ

environ["OPENCV_IO_ENABLE_JASPER"] = "true"

environ['OPENCV_IO_ENABLE_JASPER'] = '1'

import cv2

import numpy as np

from PIL import Image

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

from skimage.metrics import structural_similarity as ssim

directory = r'/*path*/'

os.chdir(directory)

I1 = cv2.imread('input image path',0)

I2 = cv2.imread('input image path',0)

I2 = cv2.resize(I2,I1.shape)

import matplotlib.pyplot as plt

I1.shape

I2.shape

import numpy as np
```

```
from sklearn.decomposition import PCA

pca_10980= PCA(n_components=10980)

pca_10980.fit(I1)

plt.grid()

plt.plot(np.cumsum(pca_10980.explained_variance_ratio_ * 100))

plt.xlabel('Number of components')

plt.ylabel('Explained variance')

plt.savefig('Scree plot.png')

import numpy as np

from sklearn.decomposition import PCA

pca_10980_= PCA(n_components=10980)

pca_10980_.fit(I2)

plt.grid()

plt.plot(np.cumsum(pca_10980_.explained_variance_ratio_ * 100))

plt.xlabel('Number of components')

plt.ylabel('Explained variance')

plt.savefig('Scree plot.png')

from sklearn.decomposition import PCA

pca =PCA(2000) # we need 200 principal components.

pca_=PCA(2000)

pca.fit(I2)

pca_.fit(I2)

converted_data = pca.transform(I1)

converted_data_ = pca.fit_transform(I2)

print(converted_data.shape)
print(converted_data_.shape)

i1 = pca.inverse_transform(converted_data)

plt.figure(figsize = (20,10))

plt.imshow(i1)

plt.savefig('pcatranformed_img.png')

i2 = pca_.inverse_transform(converted_data_)

plt.figure(figsize = (20,10))

plt.imshow(i2)

plt.savefig('pca_tranformed_img.png')

fusion_img=i1+i2/2

plt.figure(figsize = (20,10))

plt.imshow(fusion_img)

plt.savefig('pca_fusion_img.tif')

cv2.imwrite('fusedimagepcat5.jp2',fusion_img)

#we perform analysis on input image 1 and output image.

#similarly on input image 2 and output image

#first we resize the input images with respect to output images

I1 = cv2.resize(I1,fusion_img.shape)

I2 = cv2.resize(I2,fusion_img.shape)

s1= ssim(fusion_img,I1)

s1

#mean squared error

err = np.sum((fusion_img.astype("float") - I1.astype("float")) ** 2)

err /= float(fusion_img.shape[0] * fusion_img.shape[1])

mse1=err
```

```
mse1
#for input image2 and output image
#structural similarity index
s2= ssim(fusion_img,I2)
s2
#mean squared error
err = np.sum((fusion_img.astype("float") - I2.astype("float")) ** 2)
err /= float(fusion_img.shape[0] * fusion_img.shape[1])
mse2=err
mse2
```

Wavelet based image fusion:

```
from os import environ
environ["OPENCV_IO_ENABLE_JASPER"] = "true"
environ['OPENCV_IO_ENABLE_JASPER'] = '1'
import os
import pywt
import cv2
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from skimage.metrics import structural_similarity as ssim
directory = r'/*path*/'
os.chdir(directory)
```

```
def fuseCoeff(cooef1, cooef2, method):
    if (method == 'mean'):
        cooef = (cooef1 + cooef2) / 2
    elif (method == 'min'):
        cooef = np.minimum(cooef1,cooef2)
    elif (method == 'max'):
        cooef = np.maximum(cooef1,cooef2)
    else:
        cooef = []
    return cooef
FUSION_METHOD = 'mean'
I1 = cv2.imread('input image path',0)
I2 = cv2.imread('input image path',0)
I2 = cv2.resize(I2,I1.shape)
wavelet = 'db1'
cooef1 = pywt.wavedec2(I1[:,:], wavelet)
cooef2 = pywt.wavedec2(I2[:,:], wavelet)
fusedCooef = []
for i in range(len(cooef1)-1):
    if(i == 0):
     fusedCooef.append(fuseCoeff(cooef1[0],cooef2[0],FUSION_METHOD))
    else:
        c1 = fuseCoeff(cooef1[i][0],cooef2[i][0],FUSION_METHOD)
        c2 = fuseCoeff(cooef1[i][1], cooef2[i][1], FUSION_METHOD)
        c3 = fuseCoeff(cooef1[i][2], cooef2[i][2], FUSION_METHOD)
    fusedCooef.append((c1,c2,c3))
fusedImage = pywt.waverec2(fusedCooef, wavelet)
fusedImage = np.multiply(np.divide(fusedImage -
np.min(fusedImage),(np.max(fusedImage) - np.min(fusedImage))),255)
fusedImage = fusedImage.astype(np.uint8)
plt.imshow(fusedImage)
cv2.imwrite('fusedimagewavelet.jp2',fusedImage)
cv2.imshow("output",fusedImage)
cv2.waitKey(0)
#we perform analysis on input image 1 and output image.
#similarly on input image 2 and output image
#first we resize the input images with respect to output images
I1 = cv2.resize(I1,fusion_img.shape)
I2 = cv2.resize(I2,fusion_img.shape)
s1= ssim(fusion_img,I1)
s1
#mean squared error
err = np.sum((fusion_img.astype("float") - I1.astype("float")) ** 2)
err /= float(fusion_img.shape[0] * fusion_img.shape[1])
mse1=err
mse1
#for input image2 and output image
#structural similarity index
s2= ssim(fusion_img,I2)
s2
```

```
#mean squared error

err = np.sum((fusion_img.astype("float") - I2.astype("float")) ** 2)

err /= float(fusion_img.shape[0] * fusion_img.shape[1])

mse2=err

mse2
```

Visualization:

```
<map name="MapBox">

   <XYZImage name="mapbox_satellite" driver="xyz">

<url>http://a.tiles.mapbox.com/v4/mapbox.satellite/{z}/{x}/{y}.jpg?access_token
=pk.eyJ1IjoidmlzaHdhbmF0aDAwMDAiLCJhIjoiY2wzYWNkdHhrMDQ3YTNq
bWZ5Nmtha2tydCJ9.KIBucqpp7XENlsY7xeCvjA</url>

      <profile>spherical-mercator</profile>

   </XYZImage>

   <XYZElevation name="mapbox_terrain" driver="xyz" max_data_level="14">

      <url>http://api.mapbox.com/v4/mapbox.terrain-
rgb/{z}/{x}/{y}.pngraw?access_token=pk.eyJ1IjoidmlzaHdhbmF0aDAwMDAiL
CJhIjoiY2wzYWNkdHhrMDQ3YTNqbWZ5Nmtha2tydCJ9.KIBucqpp7XENlsY
7xeCvjA</url>

      <profile>spherical-mercator</profile>

      <elevation_encoding>mapbox</elevation_encoding>

   </XYZElevation>

   <FeatureModel name="mapbox_streets">

      <XYZFeatures name="mapbox_streets">

         <url>http://[abcd].tiles.mapbox.com/v4/mapbox.mapbox-streets-
v7/{z}/{x}/{y}.vector.pbf?access_token=pk.eyJ1IjoidmlzaHdhbmF0aDAwMDAi
LCJhIjoiY2wzYWNkdHhrMDQ3YTNqbWZ5Nmtha2tydCJ9.KIBucqpp7XENls
```

```
Y7xeCvjA</url>
           <min_level>20</min_level>
           <max_level>20</max_level>
           <profile>spherical-mercator</profile>
        </XYZFeatures>
        <feature_indexing enabled="true"></feature_indexing>
        <styles>
           <library name="resources">
              <url>../data/resources/textures_us/catalog.xml</url>
           </library>
           <style type="text/css">
              hospital {
                  icon: "../data/hospital.png";
                  icon-align:    center-center;
                  icon-declutter: true;
                  text-content:  getName();
                  altitude-clamping:       terrain;
                  altitude-resolution: 0.003;
              }
              school {
                  icon: "../data/school.png";
                  icon-align:    center-center;
                  icon-declutter: true;
                  text-content:  getName();
                  altitude-clamping:       terrain;
```

```
    altitude-resolution: 0.003;
}
               forest {
  model:              "../data/tree.osg";
  model-placement:    random;
  model-density:      10000;
  model-scale:        2.5
  altitude-clamping:      terrain;
  altitude-resolution: 0.003;
}
grass {
  model:              "../data/tree.osg";
  model-placement:    random;
  model-density:      10000;
  model-scale:        3
  altitude-clamping:      terrain;
  altitude-resolution: 0.003;
}
water {
  fill:              #6BA8FF;
  render-depth-test:  true;
  altitude-clamping:  terrain-drape;
}
buildings {
 extrusion-height:     getBuildingHeight();
```

```
  extrusion-flatten:     true;
  extrusion-wall-style:  building-wall;
  extrusion-roof-style:  building-rooftop;
  altitude-clamping:     terrain;
  altitude-resolution: 0.003;
}
building-wall {
  skin-library:     resources;
  skin-tags:        building;
  skin-random-seed: 1;
  fill:             #ffffff;
}
building-rooftop {
  skin-library:     resources;
  skin-tags:        rooftop;
  skin-tiled:       true;
  skin-random-seed: 1;
  fill:             #ffff00;
}
roads {
  stroke:             #842976;
  stroke-width:       5m;
  altitude-clamping:  terrain;
  stroke-tessellation-size: 4m;
  render-order: 100;
```

```
    }
</style>
<selector name="default" expr="selectStyle()"/>
<script language="javascript">
    function selectStyle() {
        var layer = feature.properties["mvt_layer"];
        if (layer === "building") return "buildings";
        if (layer === "water") return "water";
        if (layer === "road") return "roads";
        if (layer === "landuse") {
            var cls = feature.properties["class"];
            if (cls === "grass") {
                return "forest";
            }
            else if (cls === "wood") {
                return "forest";
            }
        }
        return null;
    }
    function getName() {
        if ("name" in feature.properties) {
            return feature.properties["name"];
        }
        return "";
```

```
                }

                function getBuildingHeight() {

                    return feature.properties["height"];

                }

            </script>

        </styles>

    </FeatureModel>

<viewpoints>

<viewpoint name="Hyderabad, India">

    <x>78.4867</x>

    <y>17.3850</y>

    <heading>-30.1011</heading>

<pitch>-34.79540299384382</pitch>

        <range>78142.53643278375</range>

</viewpoint>

   <viewpoint name="Hyderabad, India">

    <x>78.4867</x>

    <y>17.3850</y>

    <heading>-30.1011</heading>

<pitch>-10.17</pitch>

        <range>2771</range>

</viewpoint>

<viewpoint name="Delhi, India">

    <x>77.1025</x>

    <y>28.7041</y>
   <heading>-30.1011</heading>
<pitch>-30.6795</pitch>
      <range>10034.5m</range>
</viewpoint>
<viewpoint name="Banglore, India">
   <x>77.5946</x>
   <y>12.9716</y>
   <heading>-30.1011</heading>
<pitch>-30.6795</pitch>
      <range>10034.5m</range>
</viewpoint>
<viewpoint name="Chennai, India">
   <x>80.2707</x>
   <y>13.0827</y>
   <heading>-30.1011</heading>
<pitch>-30.6795</pitch>
      <range>10034.5m</range>
</viewpoint>
<viewpoint name="Mumbai , India">
   <x>72.8777</x>
   <y>19.0760</y>
   <heading>-30.1011</heading>
<pitch>-30.6795</pitch>
      <range>10034.5m</range>
</viewpoint>
```

```
<viewpoint name="Mexico,USA" heading="12.4" lat="38.8982" long="-77.0365"
height="20.67" pitch="-38.216" range="5000.38m" />
</viewpoints>
</map>
```