CHANGE DETECTION USING MULTISPECTRAL IMAGES FOR AGRICULTURAL APPLICATIONS

EPICS PROJECT REPORT submitted in partial fulfillment of the Requirements

Submitted by

MATHURAJU LASYA (198W1A1293)

VEERANKI PHANITHA SAI LAKSHMI (198W1A12C5)

TAHASEEN ANJUM (198W1A12B7)

Under the Guidance of

Mr. V. RADHESYAM, M. Tech (Ph.D.)

Assistant Professor

For the award of the degree

BACHELOR OF TECHNOLOGY
IN
INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

V R SIDDHARTHA ENGINEERING COLLEGE

(AUTONOMOUS - AFFILIATED TO JNTU-K, KAKINADA)

Approved by AICTE &Accreted by NBA

KANURU, VIJAYAWADA-7

ACADEMIC YEAR

(2020-21)

V.R.SIDDHARTHA ENGINEERING COLLEGE

(Affiliated to JNTUK: Kakinada, Approved by AICTE, Autonomous)

(An ISO certified and NBA accredited institution)

Kanuru, Vijayawada – 520007

CERTIFICATE

This is to certify that this project report titled **"CHANGE DETECTION USING MULTISPECTRAL IMAGES FOR AGRICULTURAL APPLICATION"** is a bonafide record of work done by **M. Lasya (198W1A1293), V. Phanitha Sai Lakshmi (198W1A12C5) and Tahaseen Anjum (198W1A12B7)** under my guidance and supervision is submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Information Technology, **V.R. Siddhartha Engineering College** (Autonomous under JNTUK) during the year **2021-2022**.

(Mr.V. Radhesyam)	**(Dr. M. Suneetha)**
Assistant Professor	Professor & Head
Dept. of Information Technology	Dept.ofInformation Technology

EXTERNAL EXAMINER SIGNATURE

Date of examination:

ACKNOWLEDGEMENT

First and foremost, I sincerely salute our esteemed institution **V.R SIDDHARTHA ENGINEERING COLLEGE** for giving me this opportunity to fulfilling my project. I am grateful to our principal **Dr. A.V. RATNA PRASAD**, for his encouragement and support all through the way of my project

On the submission of this project report, I would like to extend my honor to **Dr. M. Suneetha**, Head of the Department, of IT for her constant motivation and support throughout my work.

I feel glad to express my deep sense of gratefulness to my project guide **Mr.V. Radhesyam**, Assistant Professor for his guidance and assistance in completing this project successfully.

I would also like to convey my sincere indebtedness to all faculty members, including supporting staff of the Department, friends, and family members who bestowed their great effort and guidance at appropriate times without which it would have been very difficult on my part to finish the project work.

DEPARTMENT OF INFORMATION TECHNOLOGY
VELAGAPUDI RAMAKRISHNA SIDDHARTHA ENGINEERING COLLEGE

PROJECT SUMMARY

S.NO	ITEM	DESCRIPTION
1	Project Title	Change Detection Using Multispectral Images for Agricultural Applications
2	Batch Names & Numbers	M. Lasya (198W1A1293) V. Phanitha Sai Lakshmi (198W1A12C5) Tahaseen Anjum (198W1A12B7)
3	Name of The Guide	V.Radhesyam
4	Name of The Mentor	Phaneendra
5	Research Group	Computer Vision
6	Application Area	Agriculture
7	Aim of the Project	The aim is to identify the changes at a particular location at two different time stamps
8	Project Outcomes	Predicted the changes in agriculture in the Vishakhapatnam area

Student Signatures:

1. M. Lasya
2. V. Phanitha Sai Lakshmi
3. Tahaseen Anjum

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Agriculture is an important sector of India. India's agriculture has grown tremendously over the last few decades. Even though the industry has played a remarkable part in the Indian economy, agriculture's contribution to the country's prosperity cannot be neglected. Change detection implies comparing two multitemporal satellite images to look for any differences between the two timestamps. We implemented two approaches for obtaining changes between two satellite photos. The first approach includes principal component analysis (PCA) and K-Means clustering and the second approach includes Multivariate Alteration Detection. Multitemporal pictures are used to create the difference image. These algorithms generate the difference image as output as to where the development or geographical change has occurred. The resulting images are quantified with proper metrics RSME and PSNR.

Keywords: Change detection, QGIS, Principal Component Analysis, K-Means, Multivariate Alteration Detection, Remote Sensing Images, Google Colab, Multispectral Images.

CHAPTER – 1

INTRODUCTION

This chapter discusses the origin of the problem, the problem description, basic definitions, and remote sensing image applications.

1.1 Origin of the Problem:

We need to identify the changes occurring around us to promote sustainable development. Change detection is the process of identifying changes in the region through different timelines to obtain land use, land cover, and urbanization changes. Many applications use remote sensing data to perform the change detection process at their core. Change detection is divided into two categories. The first is supervised change detection, and the second is unsupervised change detection. The most common type of data is geographic data, which is usually in digital (e.g., satellite imagery), analog (e.g., aerial pictures), or vector (e.g., feature maps) format. Ancillary data (such as historical, economic, and other types of data) can also be employed.

1.2 Basic definitions and Background

Remote Sensing: The acquisition of data from a distance is known as remote sensing. Researchers can "feel" facts about the Earth by using special cameras to acquire remotely sensed photos. The many types of remote sensing are determined by the energy sources utilized. The first is active remote sensing, and the second is passive remote sensing. Active remote sensing[9] makes use of a man-made energy source. It usually sends forth signals. Natural energy sources are used in passive remote sensing. Remote sensing can cover wide areas and deliver digital data[1]. It is always effective. Data extraction necessitates the assistance of a professional.

Multispectral Images: Light from frequencies other than visible light, such as infrared, can be captured via multi-spectral imaging. This may enable the extraction of additional information that the human eye's red, green, and blue receptors are unable to collect. The term "multispectral imagery" refers to images with three to ten bands. Landsat-8 produces 11 images with the following bands:

COASTAL AEROSOL in band 1 (0.43-0.45 µm)

BLUE in band 2 (0.45-0.51 µm)

GREEN in band 3 (0.53-0.59 µm)

RED in band 4 (0.64-0.67 µm)

NEAR INFRARED (NIR) in band 5 (0.85-0.88 µm)

SHORT-WAVE INFRARED (SWIR 1) in band 6 (1.57-1.65 μm)

SHORT-WAVE INFRARED (SWIR 2) in band 7 (2.11-2.29 μm)

PANCHROMATIC in band 8 (0.50-0.68 μm)

CIRRUS in band 9 (1.36-1.38 μm)

THERMAL INFRARED (TIRS 1) in band 10 (10.60-11.19 μm)

THERMAL INFRARED (TIRS 2) in band 11 (11.50-12.51 μm)

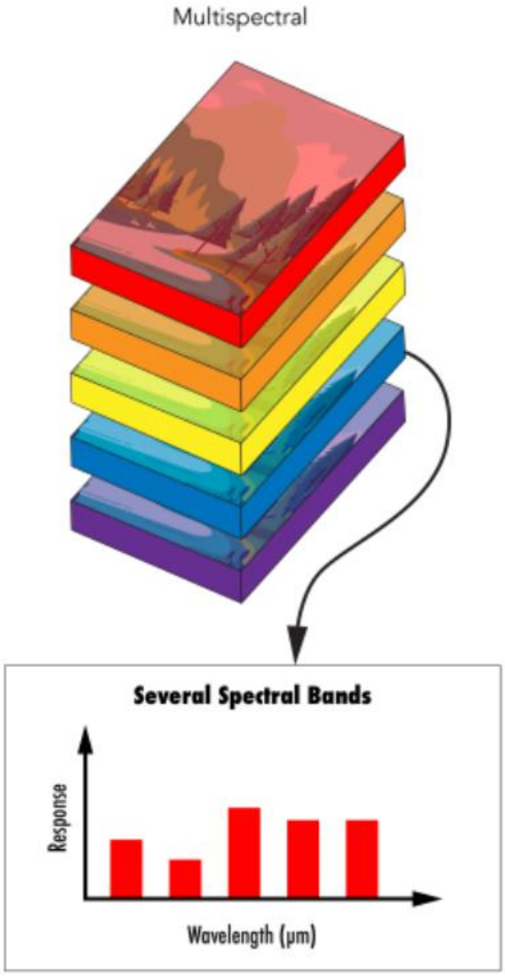

Figure 1.2: Multispectral Bands

Python: Python is a general-purpose, most widely used, and powerful programming language. Python is intended to be a very understandable language. It is a great language for programmers who are just starting. Guido van Rossum designed Python between 1985 and 1990. It is used for server-side web development, Machine Learning, Games, software development, mathematics, and system programming. Python supports the Object-Oriented style or technique of programming that encapsulates code within objects. Modules and packages are also available in the Python framework, allowing for code reusability. It also supports exception handling.

Google Colab: Google colab is a cloud service. It let us use and share our Jupiter notebook with other people and it requires absolutely no setup. You don't need to download install and run anything on your system it runs on the browser. The best part is It supports GPU and it's completely free. Your Google Colab notebooks will be saved in the Google Drive account associated with your Google account. If you want, you may also place them in a GitHub repository. You can upload notebooks from your local system to Colab, and you can also download Colab notebooks to store the notebooks locally.

1.3 Problem Statement:

Change detection captures the spatial changes from multi-temporal satellite images. Changes can be occurred due to manmade or natural phenomena. Change detection methods can be either supervised or unsupervised learning according to the nature of the data processing. In this project, we are performing Unsupervised change detection. Unsupervised change detection techniques mainly use the automatic analysis of change data which are constructed using multi-temporal images. Land use and Land cover (LULC) change detection is important in today's world because of its impacts on local climate, radiation balance, biogeochemistry, hydrology, and the diversity and abundance of terrestrial species. This mainly focused on detecting changes in the landcover.

There is a close link between Changes in land use and land cover (LULC) and population migration and economic situations. To identify the changes properly timely updating of LULC data sets are required.

The agricultural sector, as we all know, is the most important in the Indian economy. We must be aware of developments in agricultural lands (i.e., land use and land cover). This can be obtained by change detection using multi-spectral images.

1.4 Applications: Satellites have aided in the creation of several technologies, including globe mapping, GPS, and others. Remote sensing is useful in Agriculture, Forestry, Weather, Disaster Management, and Biodiversity.

Here we have seen the origin of the problem and problem statement which is taken in our project to identify the change that occurred at a particular area in two different timestamp images by using multispectral remote sensing images.

CHAPTER –2

REVIEW OF LITERATURE

This chapter describes the review of literature that we have taken from various papers and considered all the points mentioned in the papers.

2.1Description of Existing Systems:

Table 2.1: Literature Review

S.NO	AUTHOR	TITLE	JOURNAL	TECHNIQUES USED	OBSERVATION
1.	Fei song zhuoqian Yang, Yang Yang	Multi-scale feature-based land cover change detection in agriculture	IEEE Transactions on Geoscience And Remote Sensing (2018)	LULC change, multi-scale feature description, K-Means	Proposed a robust multi-temporal change detection for land cover change in agriculture
2.	Philip J. Howarth & Gregory M. Wickware	Procedures for change detection using Landsat digital data	International Journal of Remote Sensing Volume-2(1981)	Adaptive parameters, multi-temporal, remote sensing images, change detection model, superficial superposition.	Proposed procedures that might be followed in applying the complementary methods of band rationing and post-classification change detection to monitor a large remote area are outlined.
3.	Lidong Zou, Muyi Li, Sen Cao, Feng Yue, Xiufang Zhu, Yizhan Li & Zaichun Zhu	Object-Oriented Unsupervised Change Detection Based on Neighborhood Correlation Images and *k*-Means Clustering for the Multispectral and High Spatial Resolution Images	Canadian Journal of Remote Sensing	Principal Component Analysis (PCA), K-means clustering, Multivariate alteration detection (MAD).	This paper proposed the procedures used for obtaining changes in multispectral images.

4.	Prashanth R. Marpu; Paolo Gamba; Morto n J. Canty	Improving change detection results of IR-MAD algorithm, by eliminating strong changes.	IEEE Transactions on Geoscience And Remote Sensing (2011)	MAD algorithm and K-means.	This paper proposed the prior elimination of strong changes in the results of change detection in multispectral images.
5.	Sudipan Saha; Yady Tatiana Solano-Correa; Frances ca Bovolo; Lorenz o Bruzzone	Unsupervised Deep Transfer Learning-Based Change Detection for HR Multispectral Images	IEEE Transactions on Geoscience And Remote Sensing (2020)	Adaptive parameters, multi-temporal, remote sensing images, change detection model, superficial superposition.	Proposed the techniques used for unsupervised change detection.

2.2Summary of Literature Study: There are lots of change detection techniques met in the above papers. It is possible to group these techniques under two main topics as supervised and unsupervised change detection [8]. In this study, the aim is to define the land cover changes occurring in specific areas with unsupervised change detection techniques by using Landsat images belonging to different years which are obtained by the technique of remote sensing. While that process is being made, the image differencing method is going to be applied to the images by following the procedure of image enhancement. After that, the method of Principal Component Analysis is going to be applied to the difference image obtained.

The high-resolution image is referred to as high-dimensional data space. The lengthy image uploading and downloading time have always been a major issue for Internet users. Apart from the data transmission problem, high-resolution image consumes greater storage space. Principal Component Analysis (PCA) is a mathematical technique to reduce the dimensionality of data. This paper aims to evaluate the application of PCA on digital image feature reduction and compare the quality of the feature-reduced images with difference variance values. Experimental results showed that the PCA technique effectively reduces the dimension of image data while still maintaining the principal properties of the original image. This technique achieved a 35.3% for the file size reduction for the best feature reduced quality.

CHAPTER-3

PROPOSED METHOD

This chapter describes the architecture diagram and algorithms that we have applied for our project to find the change that happened between two different timestamp images.

3.1 Design Methodology: Change Detection has made extensive use of visual acuity, which is associated with the development of airborne and space sensors as well as their long-term availability and repetitive vision. Landsat images also contributed to the overcrowding of the transition sensor starting with multispectral (MSS), the thematic mapper (TM), Egrated thematic mapper (ETM), and the latest sensor, Operational Land Imager (OLI) [4].

Spectral, location, and temporary and image comparisons are important in Change Detection. Impartiality or false discovery should not obscure the real events of the transition. Advanced processing processes are essential to generate consistent data and avoid such errors. Borrowing an idea from data mining, pre-processing involves data integration, refinement, generalization, and modification.

Data integration is an important step in identifying reuse and inconsistencies while integrating and editing data sets. Integrating remote sensor data is related to image alignment by resizing and registering using the same reference and control points. Additionally, it involves rejecting and resuming images from a variety of formats into standard formats to allow comparisons.

Data cleaning is an important process for reducing noise and managing lost data and includes atmospheric adjustment and cloud coverage of visual images or filtering microwave data dots. Various pre-processing techniques and techniques are used to assess image quality, address errors, and deal with noise that causes data loss or that makes it difficult to detect and interpret.

Normalization is necessary to produce consistent but comparable data sets, especially, where long-term sequence monitoring is involved. Currently, transformation is used to improve data quality or highlight features by reducing the size or integration of frequently viewed image layers to represent biophysical features.

Binary change maps are bitemporal CD synthesis products that highlight the location of changes and are thus standard information for decision-making systems.

Image differencing investigates the dissimilarity of image properties on a pixel-by-pixel basis by simply subtracting one digital image value from another to yield a numerical difference between pixel pairs to minimize errors, differencing is typically performed on spectrally and atmospherically pre-processed images.

A covariance or correlation matrix is used in the principal component analysis for bitemporal Change Detection. PCA transforms the original matrix into new linear forms as a redundancy reduction technique, and it has been used to detect the change in Landsat data. The change information is primarily interpreted from the third and fourth principal components, which represent unique phenomena with small variances, whereas the first and second principal components generally represent variances of unchanged pixels that are larger than those of changed pixels.

The MAD algorithm has been considered the most advanced change detection algorithm for multi-spectral images due to its excellent change detection accuracy and varying stability. The main idea of MAD is to assign an initial value of 1 to each pixel in the remote sensing images. During each iteration, a new weight is assigned to the two images by calculating the chi-square distribution probability. In the next iteration, weights are considered in the calculation of the mean and variance. Through these calculations, the pixels that do not change have large weights, and the weight of each pixel tends to be stable after iterative convergence. At this time, accurate change detection results can be obtained by comparing the weight and threshold of each pixel. Another theoretical explanation for MAD was proposed in the literature: multivariate change detection is the feature that seeks the strongest image correlation. This algorithm makes full use of the band with a strong correlation, while the band with a weaker correlation is given a smaller weight, which contributes less to the calculation of variation strength. Therefore, the MAD algorithm fails to make full use of the various characteristics of each band, resulting in the incomplete detection of the details of changed areas. Therefore, the algorithm has broken patches, much noise, and small change areas that are difficult to detect, and the overall detection rate is low. In recent years, with the development of machine learning methods, neural networks have also been applied to change detection in multispectral images.

3.2 System Architecture Diagram:

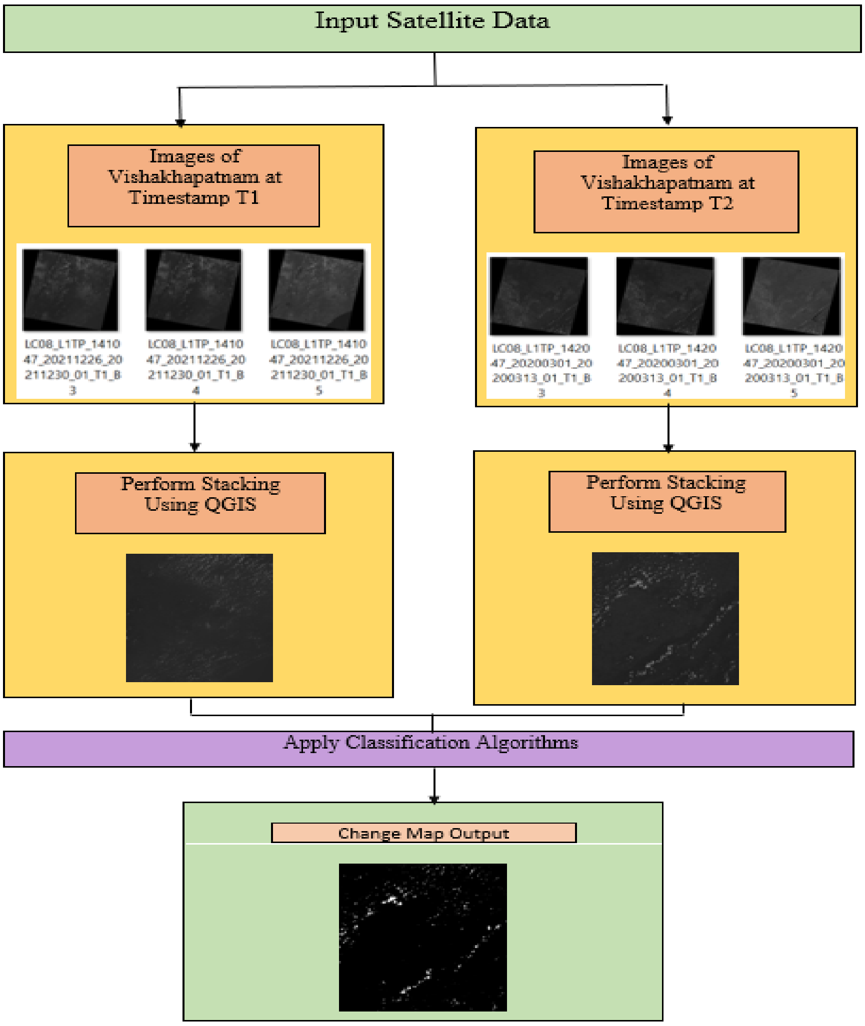

Figure 3.2: Architecture Diagram

3.3Description of Algorithms:

Algorithm: Principal Component Analysis (PCA)

Input: nxn dimensional matrix

Output: Reduced Dimension matrix of order<n

Method:

1. Begin
2. Take the 2-dimensional matrix of independent variables X. Rows represent data items and columns represent features. The number of columns is the number of dimensions.
3. For each column, subtract the mean of that column from each entry.
   //This ensures that each column has a mean of zero
4. Given the columns of X, are features with higher variance more important than features with lower variance

5. if features are independent of the variance of features, then divide each observation in a column by that column's standard deviation. Call the centered and standardized matrix Z.
6. Endif
7. Covariance of Z = Z^TZ. The resulting matrix is the covariance matrix of Z, up to a constant.
8. Calculate the eigenvectors and their corresponding eigenvalues of Z^TZ.
9. The eigen decomposition of Z^TZ is where we decompose Z^TZ into PDP^{-1},

   where P is the matrix of eigenvectors. D is the diagonal matrix with eigenvalues on the diagonal and values of zero everywhere else.
10. Take the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda p$ and sort them from largest to smallest. In doing so, sort the eigenvectors in P accordingly. Call this sorted matrix of eigenvectors P*
11. Calculate Z* = ZP*. //This new matrix, Z*, is a centered/standardized version of X but now each observation is a combination of the original variables, where the weights are determined by the eigenvector
12. We need to determine which features from the new set we wish to keep for further study. There are 3 methods: -
    - Arbitrarily select how many dimensions we want to keep
    - Calculate the proportion of variance for each feature, pick a threshold, and add features until you hit that threshold
    - Calculate the proportion of variance for each feature, sort features by the proportion of variance, and plot the cumulative proportion of variance explained as you keep more features. One can pick how many features to include by identifying the point where adding a new feature has a significant drop in variance explained relative to the previous feature, and choosing features up until that point.
13. End

Description of PCA:

PCA is a dimensionality-reduction approach for reducing the dimensionality of large data sets by transforming a large collection of variables into a smaller one that retains the majority of the information in the large set. The steps involved in PCA are:

- Standardization

This phase is used to normalize the range of continuous beginning variables so that they all contribute equally to the analysis. This can be accomplished mathematically by subtracting the mean and dividing by the standard deviation for each value of each variable.

$$Z = \frac{value - mean}{standard\ deviation} \tag{1}$$

- Covariance Matrix Computation

  The goal of this step is to understand how the variables in the input data set differ from the mean about each other, or to see if there is any relationship between them open

$$\begin{pmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{pmatrix} \tag{2}$$

- Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components

  Principal components are new variables that are created linearly by combining or combining the initial variables. So, 10-dimensional data gives you ten principal components, but PCA tries to put as much information as possible in the first component, then as little as possible in the second, and so on.
- Feature Vector

  We can find the principal components in order of significance by computing the eigenvectors and ordering them by their eigenvalues in descending order. In this step, we decide whether to keep all of these components or discard those with less significance (low eigenvalues), and then combine the remaining ones to form a matrix of vectors known as the Feature vector.

Algorithm: K-Means Clustering

Input: k: the number of clusters and D: a data set containing n objects

Output: A set of k clusters

Method:

1. arbitrarily choose k objects from D as the initial cluster centers;
2. **repeat**
3. (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
4. update the cluster means, that is, calculate the mean value of the objects for each cluster;
5. **until** no change;

K-means clustering is a straightforward and elegant method for dividing a data set into K distinct, non-overlapping clusters. To perform K-means clustering, we must first specify the desired number of clusters K; the K-means the algorithm will then assign each observation to one of the K clusters.

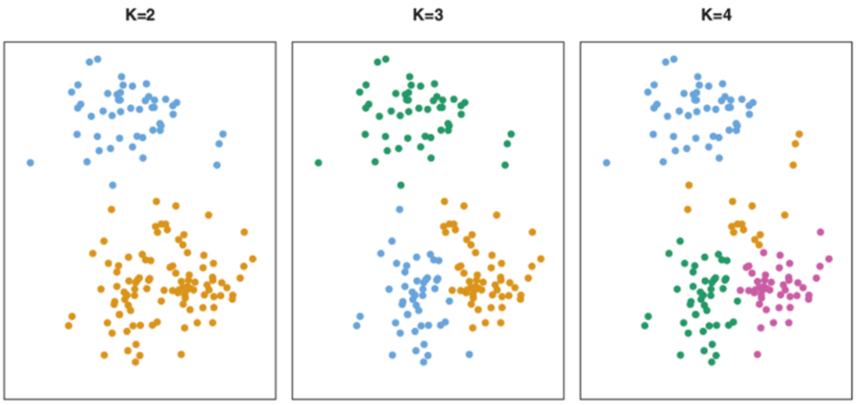

Figure 3.3: K-Means Clustering Diagram

The K-means algorithm aims to choose a centroid that minimizes the inertia, or within-cluster sum-of-squares criterion

$$\sum_{i=0}^{n} min\,(||x_i - \mu_j||)(||x_i - \mu_j||) \qquad (3)$$

- Determine the value of K, the number of clusters
- Randomly assign K distinct centroid
- Calculate the Euclidean distance between each point and centroid
- Assign each point to the nearest cluster
- Calculate the mean of each cluster as new centroid. The position of the new centroid (X,Y) after the assignment of the new point is:

$$X = (x_1 + x_2 + x_3 + \ldots\ldots xn) \qquad (4)$$

$$Y = \frac{\sum^{n} (y_1 + y_2 + y_3 + \ldots\ldots\ldots y_n)}{n} \quad (5)$$

Algorithm: Multivariate Alteration Detection

Input: Two multivariate images written as vectors (without loss of generality). The expectation values E{X} = E{Y} = 0, X = $[X_1 \cdot \cdots X_k]^T$ and Y = $[Y_1 \cdot \cdots Y_k]^T$, where k is the number of spectral bands.

Output: Label 'change' if change at a pixel i >99% and label 'no change' if change at a pixel i <1%

Method:

1. Begin
2. Apply linear transformation on image data to maximize a measure of change in simple multispectral difference image. The variance is

$$V\{v(X_1 - Y_1) + \cdots + v_k(X_k - Y_k)\} = V\{v^T(X - Y)\} \quad (6)$$

3. Maximize the variance under constrained amounts to finding principal components of the simple difference images
4. Extract the linear combinations of different coefficients for X and Y

$$a^T X = a_1 X + \cdots + a_P X_{P\cdot} \quad (7)$$

$$b^T Y = b_1 Y_1 + \cdots + bqYq \quad (8)$$

5. Maximize the variance V{a^T X – b^T Y}. Manually make choices for a and b requesting unit variance for a^{T}X and b^T Y.

$$V\{a^T X\} = V\{b^T Y\} = 1 \quad (9)$$

6. We need to find the relationship of two sets of linear combinations. This is called canonical correlation analysis. Denote the variance-covariance matrix of X, Y and variance between them.

$$\Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} a = \rho^2 \Sigma_{11} a \quad (10)$$

$$\Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12} a = \rho^2 \Sigma_{22} a \quad (11)$$

ρ = canonical correlation of a^{T}X and b^T Y.

7. To maximize the variance in (4), we must minimize the r.
8. Define MAD variates

   MAD variate 1 = difference between highest order canonical variates.

   MAD variate 2 = difference between second highest order canonial variates
9. Dispersion matrix of MAD variates = D{a^T X – b^T Y } = 2(I-R)

   I = pxp unit matrix

   R = pxp matrix containing ascendingly sorted canonical correlations on the diagonal and zeroes off the diagonal.
10. After the achievement of unit variance, the MAD of pixel j follows X^2 distribution with p degrees of freedom. This step is used to assign labels “change” or “no change” in each observation by means of percentiles in the X^2 distribution.
11. If percentile > 99, label is “change”

    If percentile < 1, label is “ no change”
12. End

Description of MAD:

If we have two multivariate images with variables at a given location written as vectors (we assume that the expectation values, E{X} = E{Y} = 0, X = $[X_1 \cdot \cdots X_k]^T$ and Y = $[Y_1 \cdot \cdots Y_k]^T$,where k is the number of spectral bands, then the vector of band-wise differences, also known as the change vector, is a simple spectral change detection transformation. Simple differences, in general, make sense only when the data is normalized to a common zero and scaled or calibrated over time. It is difficult to visualize change in all bands simultaneously if our image data contains (many) more than three spectral bands. To address this issue and focus information on the change, image data linear transformations that optimize some measure of change (also known as a design criterion) can be considered. A linear transformation that maximizes deviations from no change, such as the variance, will maximize a measure of change in the simple multispectral difference image.

$$V\{v(X_1 - Y_1) + \cdots + v_k(X_k - Y_k)\} = V\{v^T(X - Y)\} \qquad (12)$$

A more parameter rich measure of change that allows different coefficients for X and Y and different numbers of spectral bands in the two sets, p, and q, respectively, where p<q, are linear combinations

$$a^T X = a_1 X + \cdots + a_P X_P. \qquad (13)$$

$$b^T Y = b_1 Y_1 + \cdots + bqYq \tag{14}$$

The multivariate alteration detection (MAD) transformation as.

$$\begin{bmatrix} X \\ Y \end{bmatrix} \longrightarrow \begin{bmatrix} a_p^T X - b_p^T Y \\ a_1^T X - b_1^T Y \end{bmatrix} \tag{15}$$

The MAD transformation has the very important property that if we consider linear combinations of two sets (of p variables) and (of q variables,p<=q) that are positively correlated then the pth difference shows maximum variance among such variables. The(p,j)th difference shows maximum variance subject to the constraint that this difference is uncorrelated with the previous j ones. In this way, we sequentially extract uncorrelated difference images where each new image shows maximum difference (change) under the constraint of being uncorrelated with the previous ones. If p<q, then the projection of on the eigenvectors corresponding to the eigenvalues 0 will be in- dependent of . That part may be considered the extreme case of multivariate change detection. Because the MAD variates are linear combinations of the measured variables, they will have approximately a Gaussian distribution because of the Central Limit Theorem. In addition, if there is no change at pixel , then the ith MAD value, MAD ij, has mean 0. Assuming also independence of the orthogonal MAD variates we may expect that the sum of the squared MAD variates for pixel j after standardization to unit variance approximately follows a distribution with degrees of freedom, i.e., approximately

$$T_j = \sum_{i=1}^{p} \left(\frac{MAD_{ij}}{\sigma MAD_i} \right) \in X^2(p) \tag{16}$$

The above equation can be used to assign labels “change” or “no-change” to each observation by means of percentiles in the X^2 distribution. We may choose to assign the label “change” to observations X^2 with values greater than, say, the 99% per- centile and similarly the label “nochange” to observations with X^2 values smaller than, say, the 1% percentile. Since the MAD transformation is invariant to linear (affine) transformations these no-change observations are suitable for carrying out an automated normalization between the two points in time.

The main feature of the MAD method is the transformation from a space where the originally measured variables are ordered by wavelength into a feature space where the transformed, orthogonal variables are ordered by similarity (as measured by linear correlation). This latter ordering is considered to be more relevant for change detection purposes. Differences between corresponding pairs of variables in this latter space, i.e., differences between the canonical variates, give us the orthogonal MAD variates which can be considered as generalized difference images well suited for change detection.

3.4 Description of datasets, Requirements, and Tools:

Description of datasets: We used USGS Earth Explorer to get our data. Data from Landsat 8 was utilized in our project. Here we are using multispectral images. Landsat is a satellite that collects pictures of the Earth in general. NASA and the United States jointly control it. This satellite carries the Thermal Infrared Sensor (TIRS) and the Operational Land Imager (OLI) instruments. For the longest time, the Landsat program has been operational. Landsat 8 is the Landsat program's eighth satellite. It was released on February 11th, 2013. Landsat 8 has a 30-meter resolution. These are passive sensors that solely measure solar energy that is reflected or emitted by the Earth. Every day, the staff records and observes Landsat's 8 scenes.

As we are using Landsat 8 satellite images it gives us a multispectral image that consists of Eleven bands.

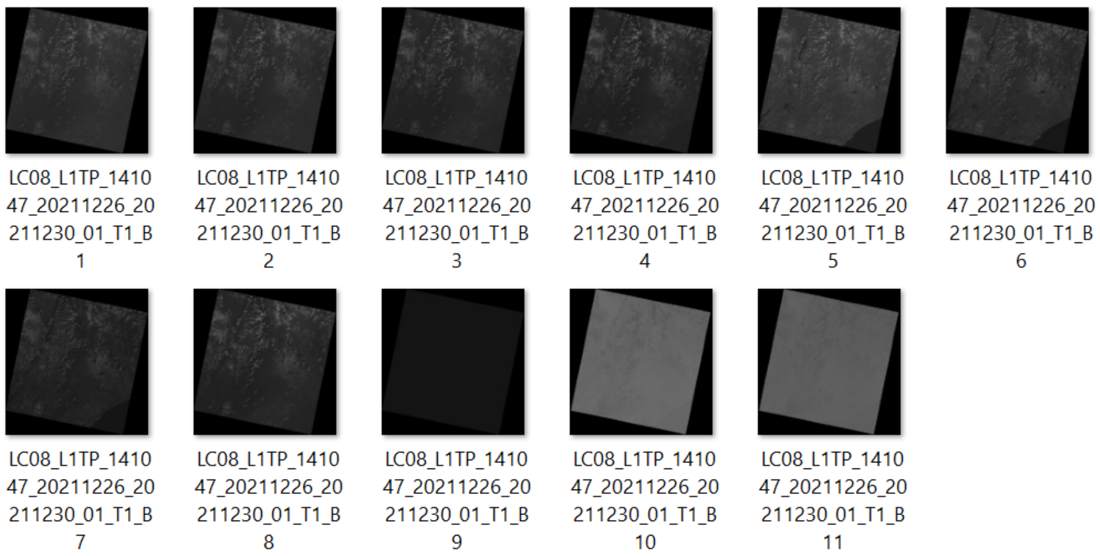

Figure 3.4.1: Bands of the Vishakhapatnam area in the year 2010

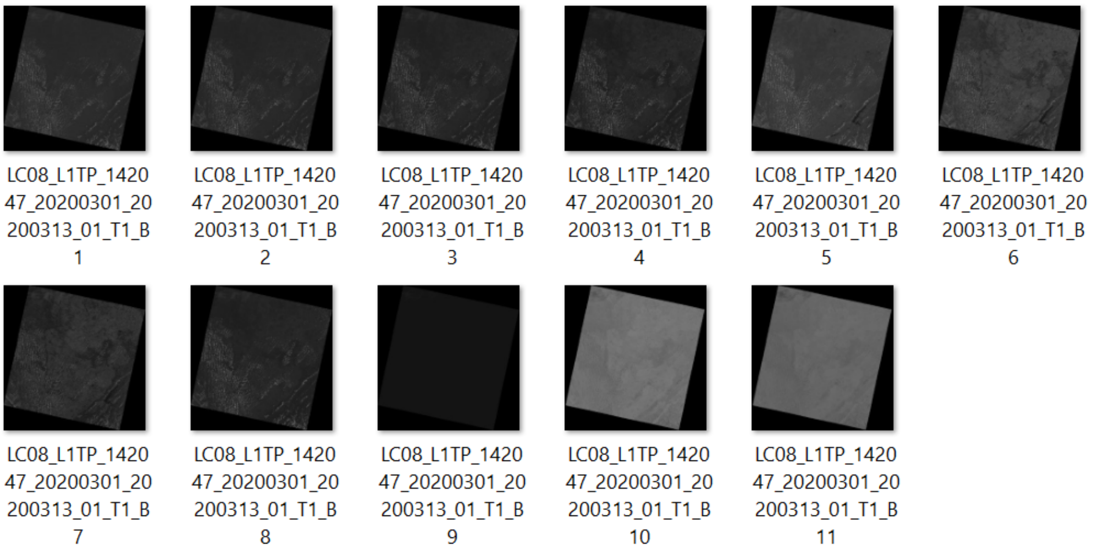

Figure 3.4.2: Bands of the Vishakhapatnam area in the year 2020

USGS EarthExplorer: Users can query, search, and order satellite pictures, aerial photographs, and cartographic products from a variety of sources using the USGS EarthExplorer (EE) tool. EE allows us to obtain satellite photos of a certain location. In EE, we can choose a location's latitudes and longitudes. We can also choose a date range.

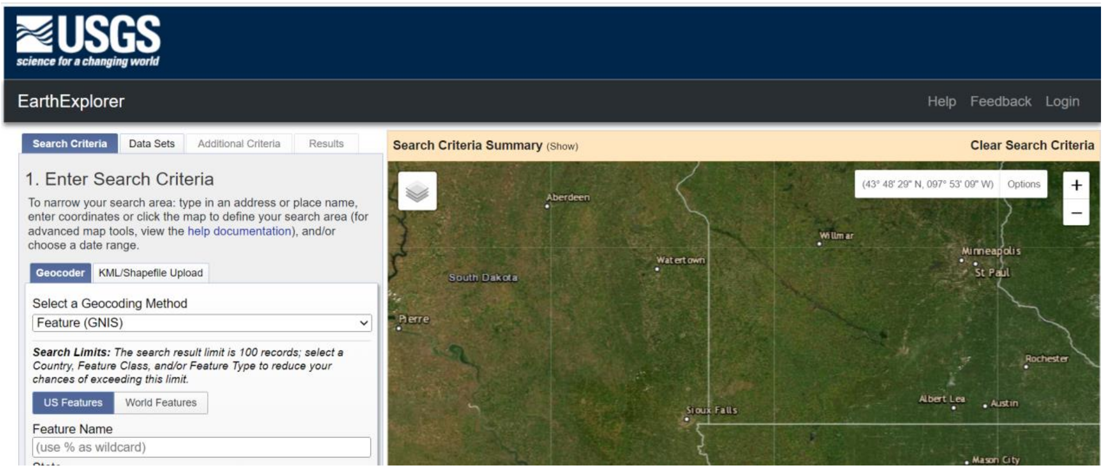

Figure 3.4.3: View of USGS EarthExplorer

QGIS: QGIS is a free and open-source cross-platform desktop geographic information system (GIS) application that supports viewing, editing, printing, and analysis of geospatial data. In our project, QGIS is primarily utilized for stacking. Our image has multiple bands because we are using multispectral imagery. We merge them by stacking them in QGIS.

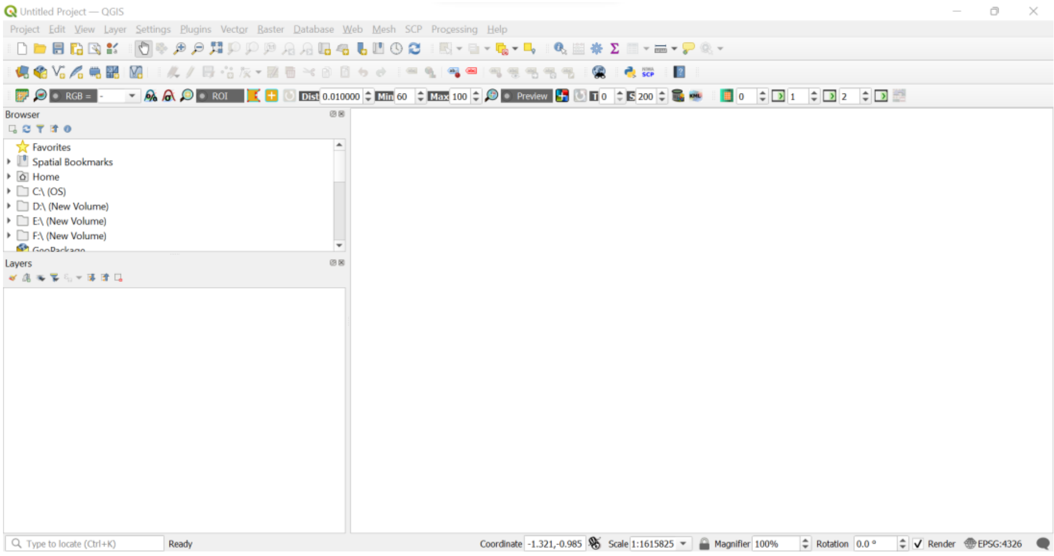

Figure 3.4.4: View of QGIS

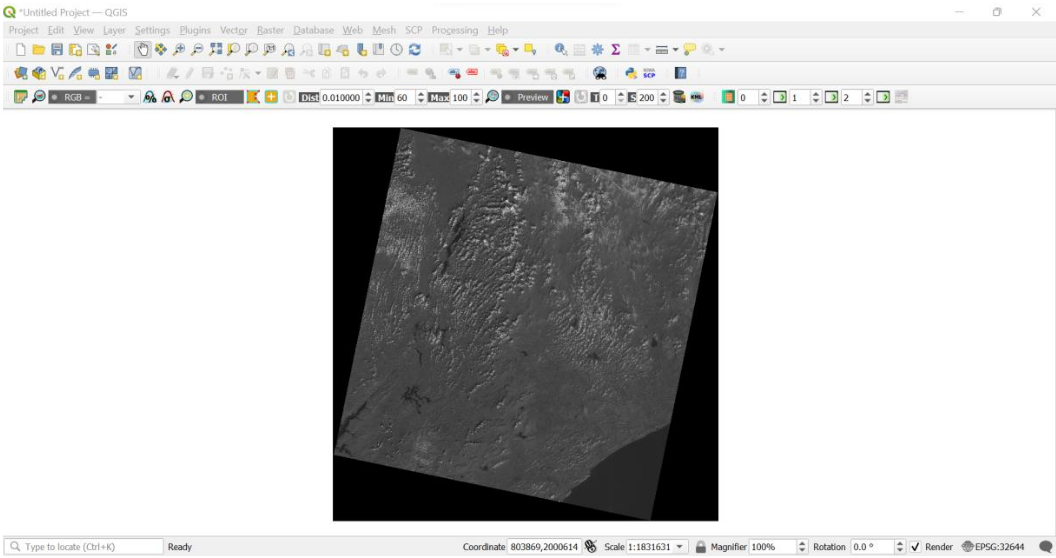

Figure 3.4.5: Opening an image in QGIS

Google Colab: Google Colab was created to give anyone who needs GPUs or TPUs to build a machine learning or deep learning model they can access to them for free. Google Colab can be thought of as a more advanced version of Jupyter Notebook. Google Colab provides tons of exciting features they are - Write and execute Python 3 code without having a local setup, save your Notebooks to Google Drive, Import Notebooks from Google Drive, Free cloud service, GPUs and TPUs.

CHAPTER-4

RESULTS AND OBSERVATIONS

This chapter describes the results of our project and comparison of both the algorithms that we have used in our project to find the change between two timestamp images.

4.1Stepwise description of Results:

We used USGS EarthExplorer to obtain Landsat 8 satellite pictures, and the area and time period we chose to identify the change in an agricultural area is indicated below.

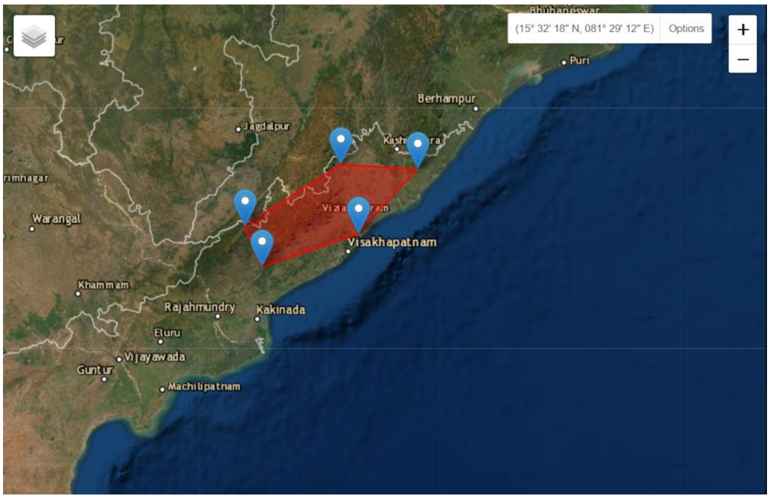

Figure 4.1.1: Visakhapatnam location

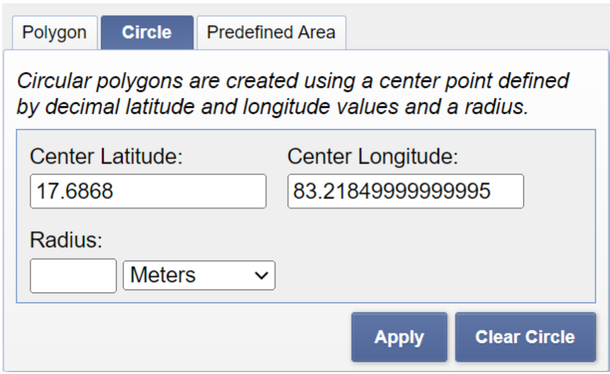

Figure 4.1.2: Latitude and Longitude values

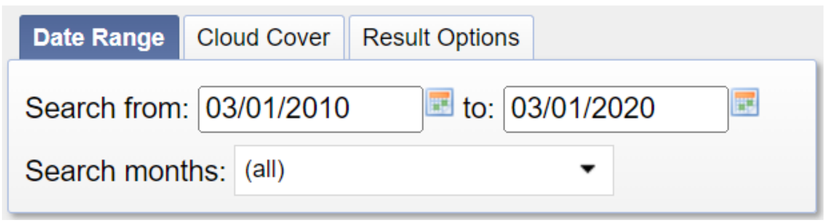

Figure 4.1.3: Time Period

Steps for obtaining the change:

Dataset-I: The location is Visakhapatnam, Andhra Pradesh and the time period is 2010.

Step 1: Combine the 3,4,5 bands among 11 bands at Time T1. We need to integrate 3,4,5 bands because we are detecting changes in agriculture.

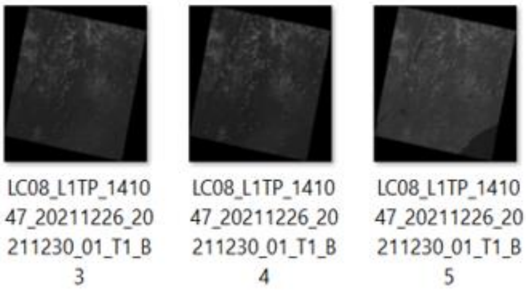

Figure 4.1.4: Bands 3,4,5 at time T1

Dataset-II: The location is Visakhapatnam, Andhra Pradesh and the time period is 2020.

Step 2: Combine the 3,4,5 bands among 11 bands at Time T2. We need to integrate 3,4,5 bands because we are detecting changes in agriculture.

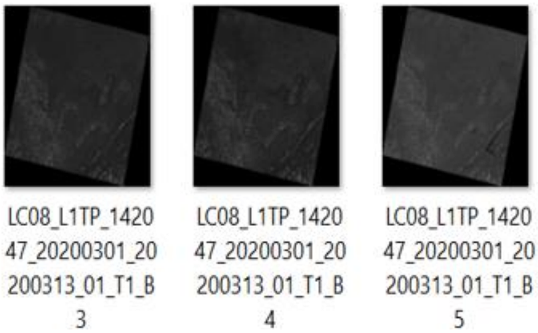

Figure 4.1.5: Bands 3,4,5 at time T2

Stacked and Clipped images:

Figure 4.1.6: Stacked Image of the year 2010 in QGIS(TI)

In the above image, we have stacked the images by using the three bands with the help of QGIS software where we will import these images in the python code of change detection and this is time stamp -1 of the 2010 year.

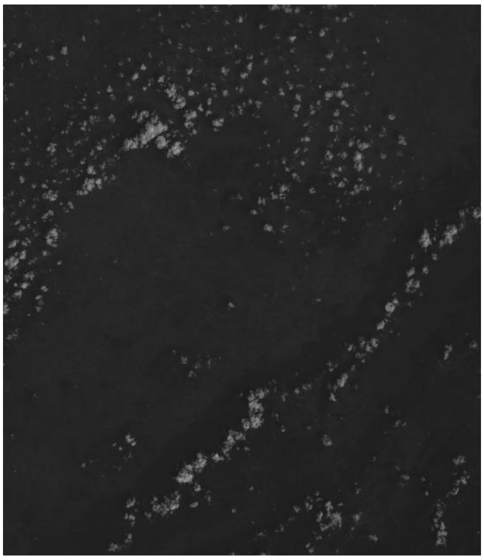

Figure 4.1.7: Stacked Image of the year 2020 in QGIS(T2)

In the above image, we have stacked the images by using the three bands with the help of QGIS software where we will import these images in the python code of change detection and this is time stamp -1 of the 2020 year.

Step 3: Install and import all the required packages. (We use cv2, NumPy, Principal Component Analysis (PCA), K-means and Multivariate Alteration Detection (MAD) modules).

- Cv2 is used for image reading
- NumPy is used for Image Matrix Processing i.e., to convert the Greyscale image into a matrix format
- PCA is used to categorize all the similar points as clusters
- We use K-Means to form super points of all the clustered points so that we can identify the dissimilar patterns
- The MAD algorithm is commonly used for this type of anomaly detection because it's highly effective and efficient. The median, or “middle” value, of all the time series at one point in time describes normal behavior for all of the time series at that timestamp.

Step 4: The next step is the generation of Difference Images. The difference image is the matrix subtraction of two images.

Step 5: Next step is the task of building the Eigen Vector Space(EVS).

We flatten non-overlapping pieces of size 5 x 5 from the difference image into row vectors using this method. scipy.misc.imresize can be used to resize the picture to make both dimensions a multiple of 5. A vector set is made up of these row vectors. Find_vector_set() in the change detection.py script performs just that. The number of rows in the vector set would be (m*n/5*5) if the size of our difference image is m x n.

Step 6: PCA is then applied to this vector set to get the Eigenvector space. We fit vector set into PCA as Training Set.

Step 7: Building the feature vector space.

Feature vector space consists of a vector set where different elements are considered as features. Function find_FVS() determines the feature vector space for us. The function is similar to find_vector_set(), but extracts overlapping blocks from the difference image.

Step 8: Clustering of the feature vector space, and change map.

The feature vectors for the pixels contain information on whether the pixels have changed or unchanged properties. A feature vector space is a set of feature vectors for every pixel. The K-means clustering algorithm gives us two clusters. One is a changed class and the other is an unchanged class. Each pixel belongs to any one of the clusters, so we can create a change map.

4.2 Test case results:

The below images are the results of our project. The result obtained in approach-1 (PCA and K-Means) and in approach-2 (MAD and K-Means) are shown below

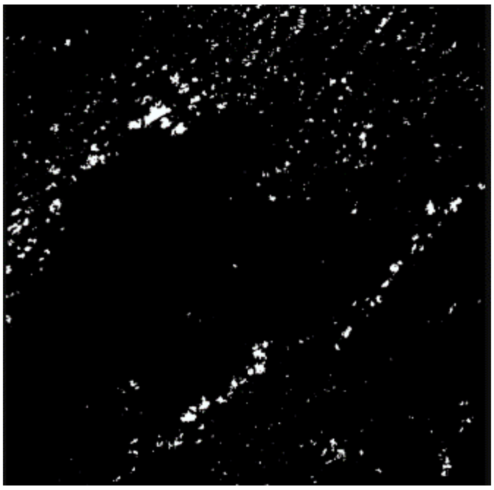

Fig. 4.1.8. Change map obtained in approach-1

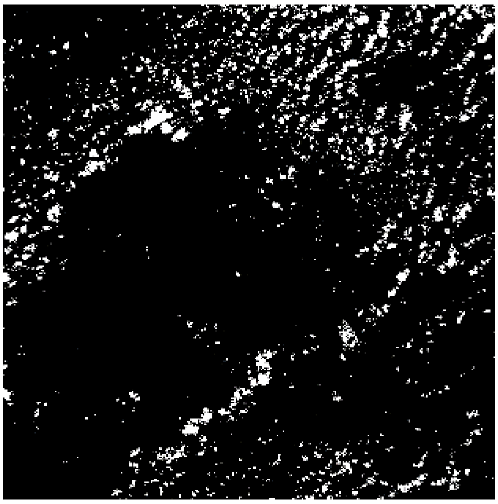

Fig. 4.1.9. Change map obtained in approach-2

After performing the two approaches i.e., change detection using PCA and K-means algorithms, MAD and K-Means algorithms, we can see the change in agricultural land in Vishakhapatnam at two separate time stamps. Our final images are grayscale images in which the white area represents the change and the black area represents the absence of change. Vishakhapatnam's change in agricultural land is represented by the white area. we can observe there is a significant decrease in agricultural land in the Vishakhapatnam area. the decrease in agriculture is due to the increase in urbanization , disasters that occurred, and industrialization. Agriculture may be a vital part of the Indian economy. Between 2010 and 2020, the modification in agricultural land use space was steadily decreasing. Temperature change, water table and quality, urbanization and industrialization, a shortage of labor and capital, and extreme weather events like floods and cyclones area unit all conducive causes. All of those parts may be detected by employing a remote sensing approach referred to as amendment detection. We are able to clearly see the amendments in two totally different timelines of exploitation change detection. This system aids farmers in crucial which sort of steps ought to be enforced to grow within the region wherever changes have occurred to spice up crop output.

4.3 Observations from the work:

After performing change detection using PCA and K-means algorithms, MAD and K-means, we can see the change in agricultural land in Vishakhapatnam at two separate time stamps. Our final image is a greyscale image in which the white area represents the change and the black area represents the absence of change. Vishakhapatnam's change in agricultural land is represented by the white area. we can observe there is a significant decrease in agricultural land in the Vishakhapatnam area. the decrease in agriculture is due to the increase in urbanization [2], disasters that occurred [5], and industrialization[3].

The metrics were applied to the two algorithms PCA and k-means, as well as mad and k-means. The RMSE and PSNR were determined. The RMSE is used to calculate the difference between the source image and the segmented picture. The PSNR block computes the peak signal-to-noise ratio in decibels between two images. This ratio is used to compare the original and compressed image quality.

Table 4.3. Metrics Obtained

Algorithm\Metrics	RMSE	PSNR
PCA and K-Means	44.48006211756571	15.167495907119415
MAD and K-Means	64.15610470390139	11.986043858816778

If RMSE is higher, the produced image will be of lesser quality. If PSNR is higher, the produced image will be of higher quality.

CHAPTER-5
CONCLUSION AND FUTURE STUDY

5.1 Conclusion:

We employed two different timestamps in this project to detect changes in a specific location, namely Vishakhapatnam. The change in agricultural lands in Vishakhapatnam was determined using pictures from the Landsat 8 satellite. Damage assessment is aided by the ability to detect changes in the environment. When comparing the two approaches, PCA and K-Means algorithm provide the image with lower RMSE and higher PSNR than MAD and K-means. Both the models had determined the change in the same location at different time stamps effectively. The project's major purpose is to detect differences in nature, such as before and after natural disasters, and estimate the percentage difference (future study), so that we may alert the NGOs who will deal with the problem and address it fast.

5.2 Future Study:

The goal of the project is to use remote sensing images to detect changes in agricultural areas. As a result, there is the potential for transformation in agricultural areas. Calculating the change percentage and percentage of change in water bodies, mountains, before and after natural disasters and other characteristics will be part of our future research.

REFERENCES

[1] Singh, A. Review Article: Digital Change Detection Techniques Using Remotely-Sensed Data. Int. J. Remote Sens.1989,10, 989–1003.

[2] Ingram, K.; Knapp, E.; Robinson, J. Change Detection Technique Development for Improved Urbanized Area Delineation; CSC/TM-81/6087; NASA, Comput, Sci. Corp.: Springfield, MD, USA, 1981.

[3] Ingram, K.; Knapp, E.; Robinson, J. Change Detection Technique Development for Improved Urbanized AreaDelineation; CSC/TM-81/6087; NASA, Comput. Sci. Corp.: Springfield, MD, USA, 1981.

[4] Lu, D.; Mausel, P.; Brondízio, E.; Moran, E. Change detection techniques. Int. J. Remote Sens.2004,25,2365–2401.

[5] L. Wen, M. Matsuoka, B. Adriano, "Damage detection due to the typhoon Haiyan from high-resolution SAR images", in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 3, no. 1, pp.123-133, May 2019, DOI: 8.1207/MGRS. 2019.2931830.

[6] L. Wen, M. Matsuoka, B. Adriano, "Damage detection due to the typhoon Haiyan from high-resolution SAR images", in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 3, no. 1, pp.123-133, May 2019, doi: 8.1207/MGRS. 2019.2931830.

[7] R. J. A. Tough, D. Blacknell and S. Quegan, "A statistical description of polarimetric and interferometric synthetic aperture radar", in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 7, no. 1, pp.67-93, April 2019, doi: 6.1107/MGRS. 2019.1431430.

[8] Kevin Louis de Jong, Anna Sergeevna Bosman, "Unsupervised Change Detection in Unlabeled Optical Remote Sensing Using CNN", in IEEE Transactions on Geoscience and Remote Sensing, vol. 2, no. 1, pp.102-114, March 2019, doi: 6.1219/MGRS. 2019.1214110.

[9] F. Bovolo and L. Bruzzone, "A theoretical framework for unsupervised change detection based on change vector analysis in the polar domain", IEEE Trans. Geosci. Remote Sens., vol. 45, no. 1, pp. 218-236, Jan. 2007.

APPENDIX

```
import numpy as np
import cv2
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from collections import Counter
import skimage

def find_vector_set(diff_image, new_size):
i = 0
j = 0
vector_set = np.zeros((int(new_size[0] * new_size[1] / 25), 25))
while i < vector_set.shape[0]:
        while j < new_size[0]:
                k = 0
                while k < new_size[1]:
                block   = diff_image[j:j+5, k:k+5]
                feature = block.ravel()
                vector_set[i, :] = feature
                k = k + 5
                j = j + 5
                i = i + 1

mean_vec   = np.mean(vector_set, axis = 0)
vector_set = vector_set - mean_vec
print(np.size(vector_set,0),np.size(vector_set,1))
return vector_set, mean_vec

def find_FVS(EVS, diff_image, mean_vec, new):
        i = 2
        feature_vector_set = []
        while i < new[0] - 2:
                j = 2
```

```
            while j < new[1] - 2:
                block = diff_image[i-2:i+3, j-2:j+3]
                feature = block.flatten()
                feature_vector_set.append(feature)
                j = j+1
            i = i+1

    FVS = np.dot(feature_vector_set, EVS)
    FVS = FVS - mean_vec
    print("\nfeature vector space size", FVS.shape)
    return FVS
def clustering(FVS, components, new):
    kmeans = KMeans(components, verbose = 0)
    kmeans.fit(FVS)
    output = kmeans.predict(FVS)
    count  = Counter(output)
    least_index = min(count, key = count.get)
    change_map  = np.reshape(output,(new[0] - 4, new[1] - 4))
    return least_index, change_map


if __name__ == "__main__":
    a = "/content/gdrive/MyDrive/stacking_output1.tif"
    b = "/content/gdrive/MyDrive/stacking_output2.tif"
    img1 = cv2.imread(a)
    img2 = cv2.imread(b)
    np.shape(img1)
    np.shape(img2)
    image1=img1[:,:,1]
    image2=img2[:,:,1]
    new_size = np.asarray(img1.shape) / 5
    new_size = new_size.astype(int) * 5
    image1 = cv2.resize(image1, (new_size[1],new_size[0])).astype(np.int16)
    image2 = cv2.resize(image2, (new_size[1],new_size[0])).astype(np.int16)
```

```
        diff_image = abs(image1 - image2)
        vector_set, mean_vec = find_vector_set(diff_image, new_size)
        pca  = PCA()
        pca.fit(vector_set)
        EVS = pca.components_

FVS  = find_FVS(EVS, diff_image, mean_vec, new_size)
k = 3
least_index, change_map = clustering(FVS, k, new_size)
change_map[change_map == least_index] = 255
change_map[change_map != 255] = 0
change_map = change_map.astype(np.uint8)
kernel1    = np.asarray(((0,0,1,0,0),
                    (0,1,1,1,0),
                    (1,1,1,1,1),
                    (0,1,1,1,0),
                    (0,0,1,0,0)), dtype=np.uint8)
kernel2=np.asarray(((0,0,0,1,0,0,0),

              (0,0,1,1,1,0,0),
              (0,1,1,1,1,1,0),
              (1,1,1,1,1,1,1),
              (0,1,1,1,1,1,0),
              (0,0,1,1,1,0,0),
              (0,0,0,1,0,0,0)),dtype=np.uint8)
cleanChangeMap = cv2.erode(change_map,kernel1)
clean=cv2.dilate(cleanChangeMap,kernel1)
cleanChangeMap_New=cv2.resize(cleanChangeMap,(np.size(img1,1),np.size(img1,0))).asty
pe(np.int16)
cv2.imwrite('a_erode.png',cleanChangeMap)
cv2.imwrite('a_without.png',change_map)
cv2.imwrite('a_dilate.png',clean)
```

MAD

```
import numpy as np


def covw(center_X, center_Y, w):
        n = w.shape[1]
        sqrt_w = np.sqrt(w)
        sum_w = w.sum()
        V = np.concatenate((center_X, center_Y), axis=0)
        V = sqrt_w * V
        dis = np.dot(V, V.T) / sum_w * (n / (n - 1))

        return dis
import gdal
from osgeo import gdal
import numpy as np
from numpy.linalg import inv, eig
from scipy.stats import chi2
import time
from sklearn.cluster import KMeans
import imageio
def IRMAD(img_X, img_Y, max_iter=50, epsilon=1e-3):
        bands_count_X, num = img_X.shape

        weight = np.ones((1, num))
        can_corr = 100 * np.ones((bands_count_X, 1))
        for _iter in range(max_iter):
            mean_X = np.sum(weight * img_X, axis=1, keepdims=True) / np.sum(weight)
            mean_Y = np.sum(weight * img_Y, axis=1, keepdims=True) / np.sum(weight)

            center_X = img_X - mean_X
            center_Y = img_Y - mean_Y


            cov_XY = covw(center_X, center_Y, weight)
            size = cov_XY.shape[0]
            sigma_11 = cov_XY[0:bands_count_X, 0:bands_count_X]
            sigma_22 = cov_XY[bands_count_X:size, bands_count_X:size]
            sigma_12 = cov_XY[0:bands_count_X, bands_count_X:size]
            sigma_21 = sigma_12.T

            target_mat = np.dot(np.dot(np.dot(inv(sigma_11), sigma_12), inv(sigma_22)), si
        gma_21)
            eigenvalue, eigenvector_X = eig(target_mat)
            eigenvalue = np.sqrt(eigenvalue)

            idx = eigenvalue.argsort()
            eigenvalue = eigenvalue[idx]
```

```
        if (_iter + 1) == 1:
            print('Canonical correlations')
        print(eigenvalue)

        eigenvector_X = eigenvector_X[:, idx]

        eigenvector_Y = np.dot(np.dot(inv(sigma_22), sigma_21), eigenvector_X)
        norm_X = np.sqrt(1 / np.diag(np.dot(eigenvector_X.T, np.dot(sigma_11, eigenve
ctor_X))))
        norm_Y = np.sqrt(1 / np.diag(np.dot(eigenvector_Y.T, np.dot(sigma_22, eigenve
ctor_Y))))
        eigenvector_X = norm_X * eigenvector_X
        eigenvector_Y = norm_Y * eigenvector_Y

        mad_variates = np.dot(eigenvector_X.T, center_X) -
 np.dot(eigenvector_Y.T, center_Y)

        if np.max(np.abs(can_corr - eigenvalue)) < epsilon:
            break
        can_corr = eigenvalue

        mad_var = np.reshape(2 * (1 - can_corr), (bands_count_X, 1))
        chi_square_dis = np.sum(mad_variates * mad_variates / mad_var, axis=0, keepdi
ms=True)
        weight = 1 - chi2.cdf(chi_square_dis, bands_count_X)

    if (_iter + 1) == max_iter:
        print('the canonical correlation may not be converged')
    else:
        print('the canonical correlation is converged, the iteration is %d' % (_iter + 1))

    return mad_variates, can_corr, mad_var, eigenvector_X, eigenvector_Y, \
           sigma_11, sigma_22, sigma_12, chi_square_dis, weight
def get_binary_change_map(data):
    cluster_center = KMeans(n_clusters=2, max_iter=1500).fit(data.T).cluster_centers_
.T

    print('k-means cluster is done, the cluster center is ', cluster_center)
    dis_1 = np.linalg.norm(data - cluster_center[0, 0], axis=0, keepdims=True)
    dis_2 = np.linalg.norm(data - cluster_center[0, 1], axis=0, keepdims=True)

    bcm = np.copy(data)
    if cluster_center[0, 0] > cluster_center[0, 1]:
        bcm[dis_1 > dis_2] = 0
        bcm[dis_1 <= dis_2] = 255
    else:
        bcm[dis_1 > dis_2] = 255
        bcm[dis_1 <= dis_2] = 0
```

```
    return bcm
data_set_X = gdal.Open("stacking_output2.tif")
data_set_Y = gdal.Open("stacking_output1.tif")

img_width = data_set_X.RasterXSize
img_height = data_set_X.RasterYSize

img_X = np.reshape(data_set_X.ReadAsArray(0, 0, img_width, img_height), (-
1, img_height, img_width))
img_Y = np.reshape(data_set_Y.ReadAsArray(0, 0, img_width, img_height), (-
1, img_height, img_width))

channel, img_height, img_width = img_X.shape

tic = time.time()

img_X = np.reshape(img_X, (channel, -1))
img_Y = np.reshape(img_Y, (channel, -1))

mad, can_coo, mad_var, ev_1, ev_2, sigma_11, sigma_22, sigma_12, chi2, noc_weig
ht = IRMAD(img_X, img_Y,
                                                   max_iter=1,
                                                   epsilon=1e-3)
sqrt_chi2 = np.sqrt(chi2)

k_means_bcm = get_binary_change_map(sqrt_chi2)
k_means_bcm = np.reshape(k_means_bcm, (img_height, img_width))
imageio.imwrite('/content/sample_data/lasya.png', k_means_bcm)
toc = time.time()
print(toc - tic)
```