North Eastern Space Applications Centre

SAMAR–**S**PACE BASED **A**NALYSIS FOR **M**ONITORING OF **A**GRO-**R**ESOURCES

Ashish Patel

Supervisor: Scientist 'SD' Pradesh Jena

Agriculture and Soil Division

Government of India, Department of Space
Umiam, Meghalaya

22/06/2024

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count: 20054

Student Name: Ashish Patel

Date of Submission:

Signature:

Contents

Abstract

Land use and land cover change has become a central component in current strategies for managing natural resources and monitoring environmental changes.

Land classification and segmentation by manual means have for long been limited by inefficiencies that take valuable time and resources. As such the LCLU comes as a sophisticated answer to these challenges. This system employs cutting-edge technologies, including machine learning (ML), object-based image segmentation (OBIS) and advanced python modules to automate and enhance land classification processes.

This research paper endeavours to examine in detail the development, implementation, and potential impact of the LCLU system. Basically, this system uses ML algorithms to accomplish an efficient land cover classification. Deeper insights will be provided by algorithms such as Random Forest, Support Vector Machine (SVM), Deep Learning, K-Nearest Neighbours (KNN). Furthermore, the system incorporates different OBIS techniques such as clustering-based segmentation, neural networks, Otsu's method, Prewitt operator network region-based segmentation Robert cross operator thresholding for accurate and reliable land segmentation.

The LCLU system has been successful largely because it has managed to integrate python modules and pretrained models so smoothly that it is easy for people to understand. It allows analysts and users to interact with the system easily by enabling them to upload satellite images, monitor classification progress and access detailed analytics about land usage. By being accessible in this manner, the LCLU system becomes a powerful environmental analytical tool while also providing researchers and policy makers with an efficient way of traversing through the intricacies of contemporary land management.

The growth in need for accurate land classification and monitoring due to environmental concerns and urban development underscores the importance of innovative approaches such as the LCLU system. This demand has increased at an exponential rate as global challenges like climate change and deforestation continue rising. The LCLU system is on top of this trend, set to retain its position after changing towards more advanced needs for environmental monitoring plus land administration.

Looking forward, the potential of the LCLU system and ML-driven land classification is vast. As technology advances, possibilities for creativity in ecological monitoring are endless. By featuring augmented reality (AR) and virtual reality (VR), this approach enhances analysis experiences and provides immersive as well as interactive visualization of land data. Moreover, recent improvements in machine learning and image processing provide an opportunity to fine-tune classification algorithms so that they can be more descriptive for detailed as well as contextually aware assessments on land use.

In sum, the LCLU Classification System offers a comprehensive response to challenges inherent in conventional methods – it signifies a paradigm shift in environmental monitoring and land management. This paper shows how the LCLU system could revolutionize landscape of land classification for efficiency, accuracy, and accessibility in the digital era. The LCLU system remains at the forefront among technological solutions applied to environmental management, aiming at redefining 21st-century classifications of territories. As such, it would be safe to say that this paper underscores how environmental monitoring has shifted towards technology-based solutions through which the future direction of 21st-century land mapping is being driven by innovative approaches like this LCLU system.

Project Specification

Project Title:

SAMAR – An ML based Land Cover and Land Use Classification System

Objective:

To develop an automated land cover and land use classification system featuring object-based image segmentation (OBIS) and machine learning algorithms to provide an efficient and accurate alternative to traditional manual methods.

Scope:

- Develop an application that can automatically classify land cover and land use from satellite and UAV images.
- Implement machine learning algorithms and OBIS techniques to enhance classification accuracy.
- Ensure secure and reliable handling of environmental data to maintain data integrity and prevent loss.

System Architecture:

- Frontend: User interface developed using Python's Tkinter for ease of use and accessibility.
- Backend: Powered by Python and relevant libraries to handle data processing, image segmentation, and land classification.

Key Features:

- Automated Land Classification: Use machine learning models to classify various land cover and land use types from satellite images.
- Object-Based Image Segmentation (OBIS): Employ techniques like clustering-based segmentation, neural networks, and thresholding for precise land segmentation.
- Real-time Analysis: Provide immediate feedback and results to users upon submission of satellite images.

Technologies Used:

- Programming Languages: Python (for both frontend and backend)
- Frontend Modules: Tkinter (for GUI development)
- Machine Learning and OBIS Libraries:
- GDAL: For reading and processing geospatial data.
- OpenCV: For image processing.
- Ultralytics YOLO: For object detection.
- NumPy: For numerical computations.
- Matplotlib: For plotting and visualization.
- Pandas: For data manipulation and analysis.
- Scikit-Learn: For machine learning algorithms like Random Forest, SVM, KNN, and clustering (KMeans, Gaussian Mixture).
- TensorFlow: For neural networks and deep learning models.
- Scipy: For advanced image processing and segmentation techniques.
- Skimage: For image processing, segmentation, and feature extraction.

- PIL: For image manipulation and display.
- TiffFile: For handling TIFF image files.

Implementation Phases:

- Phase 1: Requirement analysis and system design
- Phase 2: Development of the desktop application interface using Tkinter
- Phase 3: Integration of machine learning and OBIS algorithms
- Phase 4: Testing and validation of the system
- Phase 5: Deployment and user training
- Phase 6: Monitoring and maintenance

Expected Outcomes:

- Accurate classification of land cover and land use from satellite images.
- Enhanced efficiency and reliability compared to traditional manual methods.
- User-friendly interface for easy interaction and analysis.

Constraints:

- Availability of high-quality satellite images.
- Computational resources required for processing large images.
- Ensuring data security and user privacy.

Future Enhancements:

- Integration of augmented reality (AR) and virtual reality (VR) for immersive visualization.
- Advanced natural language processing (NLP) for more detailed analysis and reporting.
- Continuous improvement of machine learning models with more extensive datasets..

This specification outlines the framework for the SAMAR – ML based Land Cover and Land Use Classification System project, highlighting its goals, architecture, features, and implementation strategy

Chapter 1: Introduction to Remote Sensing

Remote sensing forms the basis and the most critical applications of satellites in space. In this learning journey, we will understand the concept of remote sensing and many more applicable concepts. Further, we will learn how the data about the various components of space and earth is gathered remotely, how is it processed, and much more.

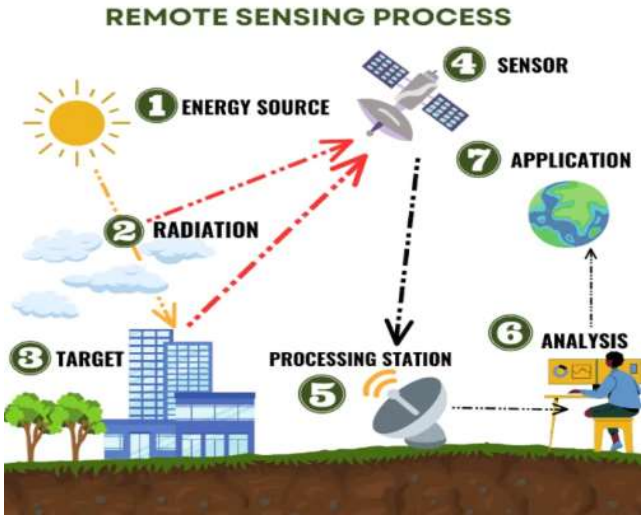


Figure 1.1: Remote Sensing Process

1.1 What is Remote Sensing?

Remote sensing is the science of acquiring information about the Earth's surface without being in direct contact with it. This process involves capturing data from a distance, typically using satellites, UAV or aircraft equipped with sensors. The sensors detect and record energy that is reflected or emitted from the Earth, which is then processed and analyzed to extract valuable information about the observed area.

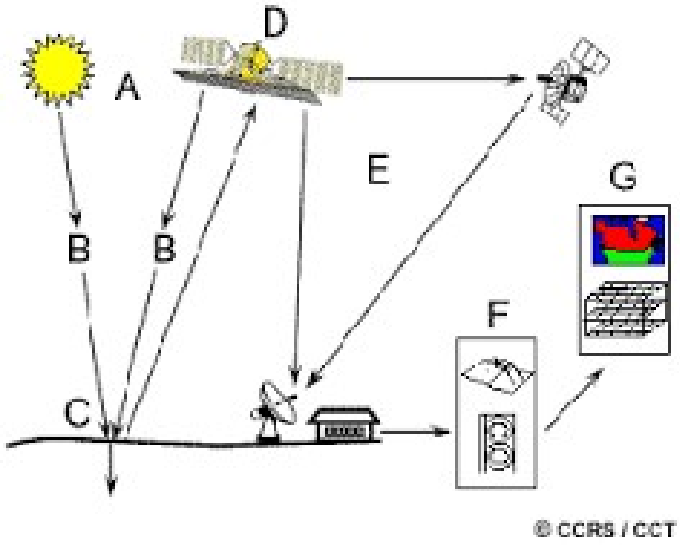


Figure 1.2 Overview of Remote Sensing 1

1.2 Historical Background

The concept of remote sensing dates back to the early 20th century with the advent of aerial photography. During World War I and World War II, aerial reconnaissance played a crucial role in military operations. However, it was the launch of the first artificial satellite, Sputnik 1, by the Soviet Union in 1957, that marked the beginning of the modern era of remote sensing. Since then, technological advancements have led to the development of sophisticated sensors and platforms, significantly enhancing our ability to monitor and study the Earth.

1.3 Principles of Remote Sensing

Remote sensing is based on the principles of electromagnetic radiation. The Sun emits energy that travels through space and interacts with the Earth's atmosphere and surface. Different materials absorb, reflect, and emit this energy differently, creating unique signatures that can be detected by sensors. The process begins with the energy source, primarily the Sun, although active remote sensing systems like radar generate their own energy and measure its reflection from the Earth's surface.

As energy travels from the source to the Earth's surface, it interacts with the atmosphere, where some of it is absorbed or scattered, while the rest reaches the surface. Upon hitting the Earth's surface, the energy is either absorbed, reflected, or transmitted, depending on the physical and chemical properties of the surface materials. Sensors on satellites or aircraft then detect the reflected or emitted energy. These sensors can operate in various parts of the electromagnetic spectrum, including visible light, infrared, and microwave. Finally, the captured data is transmitted to ground stations, where it is processed and analyzed to extract meaningful information.

1.4 Types of Remote Sensing

Remote sensing can be categorized into two main types based on the source of energy and the type of sensors used: passive and active remote sensing. Each type has unique characteristics, applications, and advantages.

Passive Remote Sensing

Passive remote sensing relies on natural energy sources, primarily sunlight, to illuminate the Earth's surface. Sensors detect and measure the energy that is naturally reflected or emitted from the surface. This method is widely used due to its ability to capture data across various spectral bands and its applicability in numerous fields.

Characteristics of Passive Remote Sensing:

- **Natural Energy Source:** Passive sensors rely on the Sun as the primary source of energy. The sensors measure the energy that is reflected or emitted from the Earth's surface.
- **Spectral Bands:** Passive sensors can capture data across multiple spectral bands, including visible light, near-infrared, and thermal infrared. This multispectral capability allows for comprehensive analysis of surface features.
- **Dependence on Daylight:** Since passive sensors rely on sunlight, data acquisition is limited to daytime conditions. Additionally, the quality of the data can be affected by atmospheric conditions such as clouds and haze.

Examples of Passive Sensors:

- **Optical Sensors:** These sensors capture visible light and near-infrared radiation, providing high-resolution images of the Earth's surface. They are used in applications such as land cover mapping, vegetation monitoring, and urban planning.
- **Thermal Sensors:** These sensors detect thermal infrared radiation emitted by the Earth's surface. They are useful for monitoring temperature variations, detecting heat sources, and studying thermal properties of materials.

Applications of Passive Remote Sensing:

- **Environmental Monitoring:** Tracking changes in vegetation, deforestation, and desertification.
- **Agriculture:** Assessing crop health, soil moisture, and pest infestations.
- **Urban Planning:** Mapping land use, infrastructure, and population growth.
- **Climate Studies:** Monitoring sea surface temperatures, snow cover, and ice extent.

Active Remote Sensing

Active remote sensing involves the use of sensors that emit their own energy towards the Earth's surface and measure the reflected or backscattered energy. This approach allows for data collection regardless of the time of day or weather conditions, making it highly versatile and reliable.

Characteristics of Active Remote Sensing:

- **Own Energy Source:** Active sensors generate their own energy, typically in the form of microwave or laser pulses. This energy is directed towards the Earth's surface and the reflected signal is measured.
- **All-weather Capability:** Since active sensors do not rely on sunlight, they can operate day and night and under various weather conditions, including cloud cover and rain.
- **High Precision:** Active sensors provide high-precision measurements of surface features, making them suitable for detailed topographic mapping and other precise applications.

Examples of Active Sensors:

- **Radar:** Radar sensors emit microwave radiation and measure the backscattered signal. They are used for applications such as surface deformation monitoring, soil moisture measurement, and sea ice tracking.
- **LIDAR (Light Detection and Ranging):** LIDAR sensors emit laser pulses and measure the time it takes for the pulses to return after reflecting off the surface. LIDAR is used for creating detailed 3D models of terrain, vegetation structure, and urban environments.

Applications of Active Remote Sensing:

- **Topographic Mapping:** Creating high-resolution digital elevation models (DEMs) and terrain maps.
- **Forestry:** Assessing forest structure, biomass, and canopy height.
- **Disaster Management:** Monitoring flood extents, landslides, and earthquake-induced ground displacement.
- **Infrastructure Monitoring:** Inspecting bridges, roads, and buildings for structural integrity.

1.5 Platforms for Remote Sensing

The vehicle or carrier for a remote sensor to collect and record energy reflected or emitted from a target or surface is called a platform. The sensor must reside on a stable platform removed from the target or surface being observed. Platforms for remote sensors may be situated on the ground, on an aircraft or balloon (or some other platform within the Earth's atmosphere), or on a spacecraft or satellite outside of the Earth's atmosphere.

Typical platforms are satellites and aircraft, but they can also include radio-controlled aeroplanes, balloons kits for low altitude remote sensing, as well as ladder trucks or 'cherry pickers' for ground investigations. The key factor for the selection of a platform is the altitude that determines the ground resolution and which is also dependent on the instantaneous field of view (IFOV) of the sensor on board the platform.

Ground Based Sensors

Ground-based sensors are often used to record detailed information about the surface which is compared with information collected from aircraft or satellite sensors. In some cases, this can be used to better characterize the target which is being imaged by these other sensors, making it possible to better understand the information in the imagery.

Ground based sensors may be placed on a ladder, scaffolding, tall building, cherry-picker, crane, etc.

Aerial Platforms

Aerial platforms are primarily stable wing aircraft, although helicopters are occasionally used. Aircraft are often used to collect very detailed images and facilitate the collection of data over virtually any portion of the Earth's surface at any time.

Satellite Platforms

In space, remote sensing is sometimes conducted from the space shuttle or, more commonly, from satellites. Satellites are objects which revolve around another object - in this case, the Earth.

For example, the moon is a natural satellite, whereas man-made satellites include those platforms launched for remote sensing, communication, and telemetry (location and navigation) purposes.

Because of their orbits, satellites permit repetitive coverage of the Earth's surface on a continuing basis. Cost is often a significant factor in choosing among the various platform options.

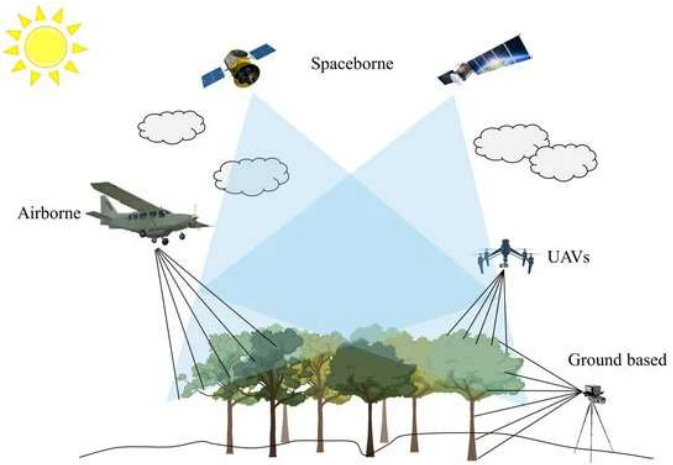


Figure 1.3 Platforms of Remote Sensing

1.6 Applications of Remote Sensing

Remote sensing has a wide range of applications across various fields. It is used for environmental monitoring to track changes such as deforestation, desertification, and climate change. Satellite imagery provides critical data for tracking changes in land cover, vegetation health, and water bodies. In agriculture, remote sensing supports precision agriculture by providing information on crop health, soil moisture, and pest infestations, enabling farmers to make informed decisions about irrigation, fertilization, and pest control. Urban planning and development benefit from remote sensing data for mapping land use, infrastructure, and population growth, aiding in efficient resource allocation and sustainable development. Remote sensing plays a vital role in disaster management by providing timely information on natural disasters such as floods, hurricanes, and earthquakes, aiding in disaster preparedness, response, and recovery efforts. In geology and mineral exploration, remote sensing techniques are used to map geological features and identify mineral resources. In hydrology, it helps study water resources, including the mapping of rivers, lakes, and groundwater, providing data on water quality, sedimentation, and hydrological cycles.

1.7 Advantages and Limitations

Remote sensing offers several advantages, including global coverage, non-intrusive data collection, temporal analysis, and multispectral imaging. Satellites provide comprehensive coverage of the Earth's surface, enabling monitoring at a global scale. Remote sensing allows data collection without physical contact, making it suitable for inaccessible or hazardous areas. It enables monitoring changes over time through repeated observations, and sensors capture data in multiple spectral bands, providing valuable information about different features and phenomena.

However, remote sensing also has limitations. The atmosphere can affect the quality of remote sensing data through scattering and absorption of electromagnetic radiation. The spatial, spectral, and temporal resolution of sensors may limit the level of detail in the data. High-resolution satellite imagery and advanced remote sensing equipment can be expensive, and analyzing remote sensing data requires specialized software and expertise.

1.8 Future Trends in Remote Sensing

The field of remote sensing is continually evolving, with advancements in technology driving new applications and capabilities. Future trends include the development of more advanced sensors with higher resolution and sensitivity, integration with other technologies like

Geographic Information Systems (GIS), artificial intelligence (AI), and machine learning for enhanced data analysis and decision-making. The use of UAVs is expected to expand for high-resolution, flexible, and cost-effective data collection. Big data techniques will be leveraged to handle and analyze the vast amounts of data generated by remote sensing platforms.

In conclusion, remote sensing is a powerful tool for observing and understanding the Earth's surface. Its diverse applications and continuous technological advancements ensure its importance in addressing global challenges and contributing to sustainable development.

Chapter 2: Introduction to GIS

The Geographic Information System (GIS) is a computer system that analyzes and displays geographically referenced information. It uses data that is attached to a unique location. If, for example, a rare plant is observed in three different places, GIS analysis might show that the plants are all on north-facing slopes that are above an elevation of 1,000 feet and that get more than ten inches of rain per year. GIS maps can then display all locations in the area that have similar conditions, so researchers know where to look for more of the rare plants. By knowing the geographic location of farms using a specific fertilizer, GIS analysis of farm locations, stream locations, elevations, and rainfall will show which streams are likely to carry that fertilizer downstream. These are just a few examples of the many uses of GIS in earth sciences, biology, resource management, and many other fields.

2.1 What is GIS?

GIS a system designed to capture, store, manipulate, analyze, manage, and present spatial or geographic data. It combines layers of information about a place to give a better understanding of that place. These layers of information can be anything from physical features like mountains, rivers, and roads to more complex data like the distribution of diseases, land use, and demographics.

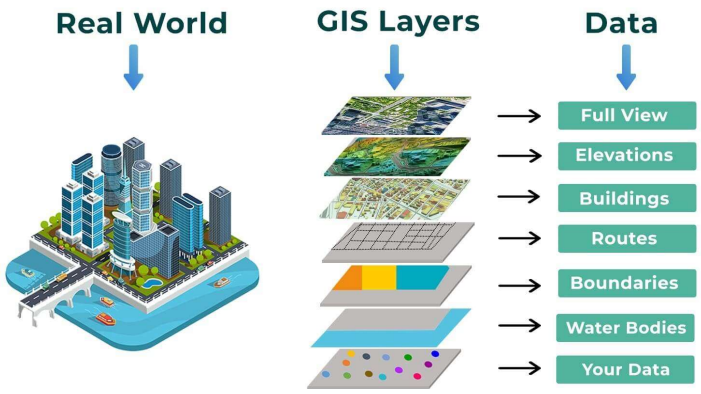


Figure 2.1 GIS Implementation

2.2 Components of GIS

GIS is composed of five key components:

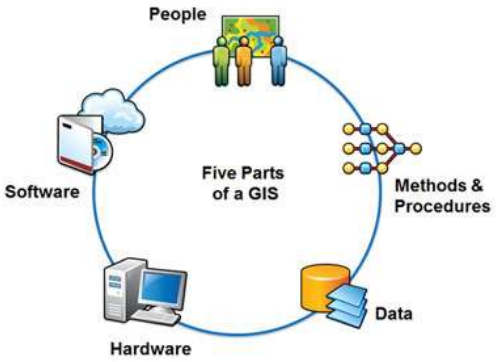


Figure 2.2 GIS Key Components

1. **Hardware**: The physical devices on which a GIS operates. This includes computers, servers, GPS devices, and other related hardware.
2. **Software**: Programs and applications used to process and analyze spatial data. Examples include ArcGIS, QGIS, and GRASS GIS.
3. **Data**: The raw information that GIS processes. This can be spatial data (maps, satellite images) or attribute data (descriptions, measurements).
4. **People**: The users who input, analyze, and interpret GIS data. They include GIS specialists, analysts, and decision-makers.

Methods: The procedures and techniques used to collect, analyze, and interpret GIS data. This involves data collection methods, data processing techniques, and analytical method

2.3 Types Of GIS Source

In Geographic Information Systems (GIS), data is categorized broadly into spatial and non-spatial data types, each serving distinct purposes and used in various applications.

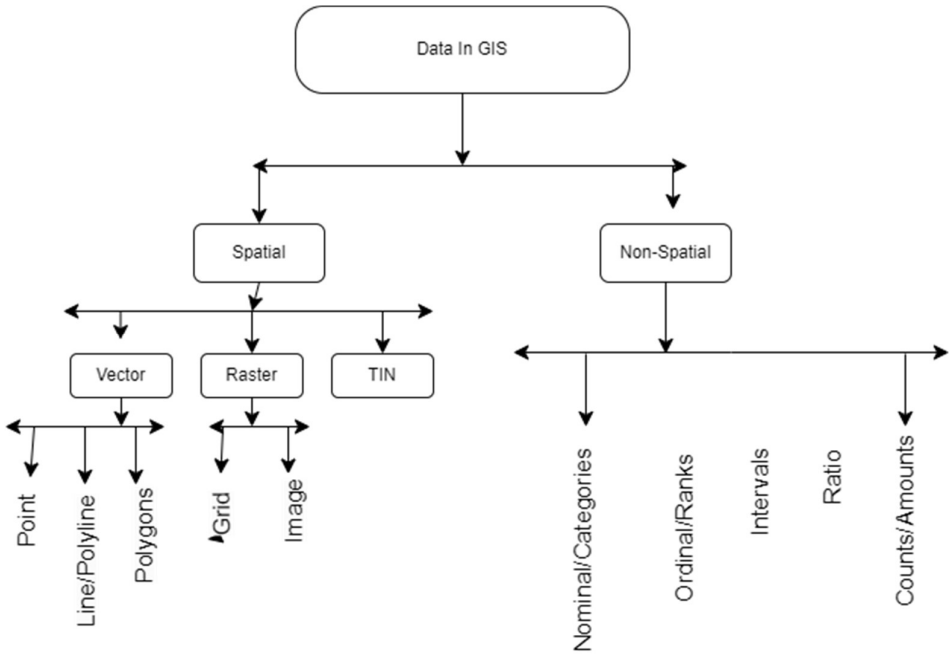
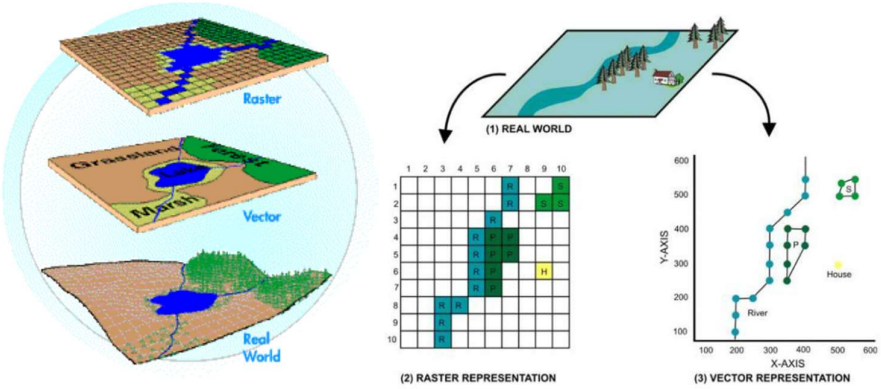


Figure 2.3 Types Of GIS

Spatial Data

Definition: Spatial data refers to information that has a direct association with geographic locations or positions on the Earth's surface. It describes where things are located and their spatial relationships

Types of Spatial Data:



Vector Data:

- Uses points, lines, and polygons to represent discrete geographic features.
- **Points:** Represent specific locations, such as cities or sampling sites.
- **Lines:** Represent linear features like roads, rivers, or pipelines.
- **Polygons:** Represent areas with defined boundaries, such as land parcels or administrative boundaries.
- Examples: Cadastral maps, road networks, land use zoning.

Raster Data:

- Composed of a grid of cells or pixels, each with a value representing a specific attribute or phenomenon.
- **Continuous Data:** Represents continuous fields like elevation models, temperature maps, and satellite imagery.
- Examples: Digital Elevation Models (DEMs), satellite imagery, climate data.

Non-Spatial Data

Definition: Non-spatial data, also known as attribute or tabular data, describes characteristics or attributes associated with spatial features. Unlike spatial data, it does not have direct geographic coordinates but is linked to spatial data through identifiers or keys.

Types of Non-Spatial Data:

1. **Tabular Data:**
   - Organized in tables with rows and columns, where each row represents a spatial feature or object, and each column represents an attribute or characteristic.
   - Examples: Population statistics, land ownership details, temperature records.
2. **Textual Data:**
   - Descriptive information associated with spatial features, often stored in narrative form or documents.

Integration of Spatial and Non-Spatial Data

GIS systems integrate spatial and non-spatial data to provide comprehensive insights and support decision-making processes. By combining spatial location with descriptive attributes, GIS users can analyze spatial patterns, relationships, and trends effectively. This integration enhances the utility and value of GIS applications across diverse fields such as environmental science, urban planning, agriculture, and public health.

2.4 Types of GIS Data Sources

GIS data can be derived from various sources, including:

Aerial Data:

- Captured from aircraft or drones, providing high-resolution images of the Earth's surface.
- Useful for detailed analysis of land use, urban planning, and environmental monitoring.

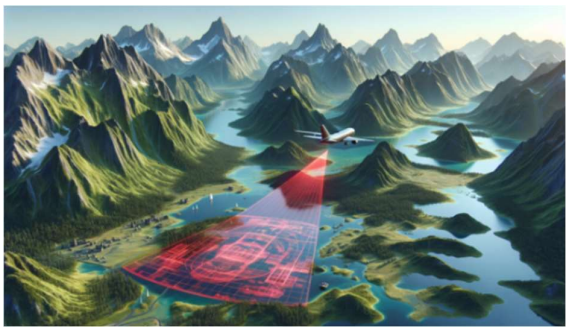


Figure 2.4 Aerial Data

Satellite Data:

- Collected from satellites orbiting the Earth, offering broad coverage for monitoring large-scale environmental changes.Used in applications such as climate change studies,

  deforestation tracking, and agricultural monitoring.

LIDAR Data:

- Uses laser pulses to measure distances to the Earth's surface, generating detailed 3D models.
- Valuable for topographic mapping, forestry, and urban planning.

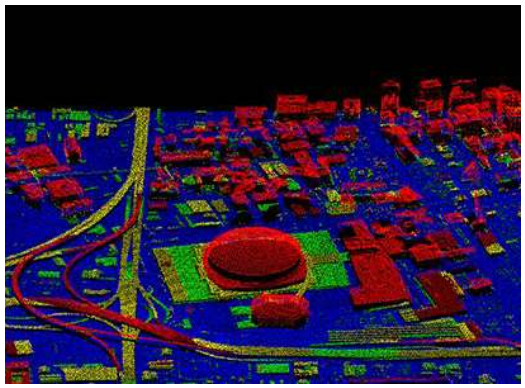


Figure 2.5 Lidar Data

Drone Data:

- Captured by UAVs equipped with cameras and sensors, providing high-resolution and flexible data collection.
- Effective for localized surveys, precision agriculture, and disaster assessment.

Ground Surveys:

- Data collected through direct measurements on the ground using tools like GPS and total stations.
- Provides highly accurate and specific information, often used to validate remote

Chapter 3: Introduction to Land Classification and Land Use(LULC)

Land cover and land use (LCLU) classification, since it can significantly contribute to environmental sustainability— urban planning and resource allocation, is more important than ever before. The manual interpretation of satellite images along with field surveys forms the basis for many traditional land classification approaches; yet such methods are laborious in terms of time and resources and unreliable owing to subjectivity. The inadequacy of these conventional methods in addressing emerging dynamics in land use underscores the need for high-tech operational systems.

The response to these challenges is what this initiative is all about: constructing SAMAR, a high-tech system for land cover and land use classification which takes full advantage of object-based image segmentation (OBIS) and machine learning algorithms. The main focus is to make a robust, efficient, and accurate classification system dashboard that can process and analyse satellite imagery with minimal human intervention, and hence enhancing the reliability and timeliness of LCLU data.

A combo of many new technologies is integrated into the SAMAR system. To this end, Python acts as the most fundamental programming language which utilizes its huge data processing, machine learning and image analysis libraries. The graphical user interface (GUI) is produced by using Tkinter, a software tool that lets users interact with the system in a friendly manner. A variety of machine learning methods such as **Random Forest**, **Support Vector Machine (SVM), K-Nearest Neighbours (KNN)** as well as **deep learning** are used for classification of land uses. Further still, **OBIS** approaches e.g. **clustering based segmentation, region-based segmentation and neural networks** have been employed to enhance the accuracy and detail of classification process.

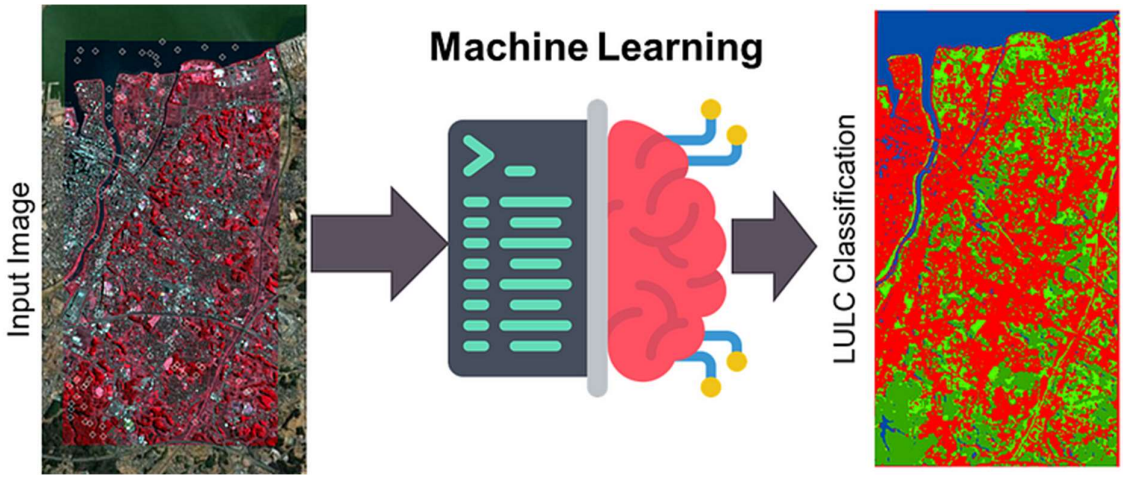


Figure 3.1 LULC System

In the real-world, SAMAR system is used to classify things like land cover in places such as Shillong and Meghalaya. These areas are characterized by diverse and complicated landscapes and as a result provide an ideal platform for assessing how well the system functions. By accurately classifying different types of lands covered in these areas, the system shows its capacity to enhance environmental monitoring efforts, urban planning practices and resource management attempts.

Moreover, GIS is also incorporated into this system leading to improved functionality which aids in visualization and spatial data analysis. GIS plays a significant role in managing, analyzing and representing geographic information; thus facilitates decision making processes relating to agriculture, forestry, urban development among others.

3.1 Purpose of Report

The purpose of this report is to furnish a comprehensive and detailed documentation of the design, implementation, and potential impacts of the SAMAR system. Being an advanced land cover and land use (LCLU) classification system, it improves the quality of land classification process by using cutting edge **machine learning algorithms, object based image segmentation (OBIS) techniques and Geographic Information Systems (GIS).** The report's objective is based on explaining the various aspects of constructing such architecture including components, methods utilized as well as technologies that were used in its creation.

Another purpose is to give an account about the main reasons for developing a modern LCLU classification system. It seeks to outline shortcomings, problems, and limitations inherent in conventional land classification approaches which will set up a strong argument for automation as well as advanced technologies in this regard. Through these pain points identified by the report, it aims at providing a clear justification for moving towards a more contemporary technology-based method of analyzing land use.

Furthermore, this paper aims at identifying the objectives and the purposes of the SAMAR system. Regarding this aspect, the system explains how the proposed approach will improve the classification of land while using satellite imagery through automation and implementation of improved OBIS and machine learning algorithms. The report also describes how the system meets the demand for timely and accurate LCLU information which is important in administration of the environment, planning of towns, and distribution of resources.

Moreover, this report acquaints the reader with the technological framework of the SAMAR system. It also elaborates how to choose and incorporate such technologies based on the best practice as programming language Python, tool for GUI development TKinter, as well as Random Forest, Support Vector Machine (SVM), K-Nearest Neighbours (KNN), and deep learning methods for the LCLU classification. The format of report also focuses on the types of OBIS techniques such as clustering based segmentation, region based segmentation and GIS for spatial analysis and visualization. Said information outlines and describes the technological features of the system and ties to give insightful details towards the architecture and adaptation of the system in an effort to facilitate the process of reference and/or future innovations as they relate to this category of systems.

Finally, it is the authors' intent to depict how the use of the SAMAR system has the possibility to revolutionize the aspects of land classification and consequently, the management of the environment. Therefore, this paper seeks to show that with such a system it becomes easy to carry out LCLU classification with much less effort and time as compared to the traditional methods; that the results are accurate and consistent; and that the efficiency of the land use analysis is improved. In this respect, the following report aims at providing a further insight into the case and thus to make a certain Academic contribution to the discussion on the utilization of sophisticated technologies in the sphere of environmental control and rational utilization of natural resources.

3.2 Overview Of Automated Land Cover and Land Use

The SAMAR system is an effective solution designed to revolutionize the traditional processes of land cover and land use (LCLU) classification by integrating advanced machine learning algorithms, object-based image segmentation (OBIS) techniques, and cutting-edge Geographic Information Systems (GIS). This system aims to address the inefficiencies and challenges associated with conventional LCLU methods, providing a more efficient, accurate, and scalable alternative.

Core Components and Functionalities

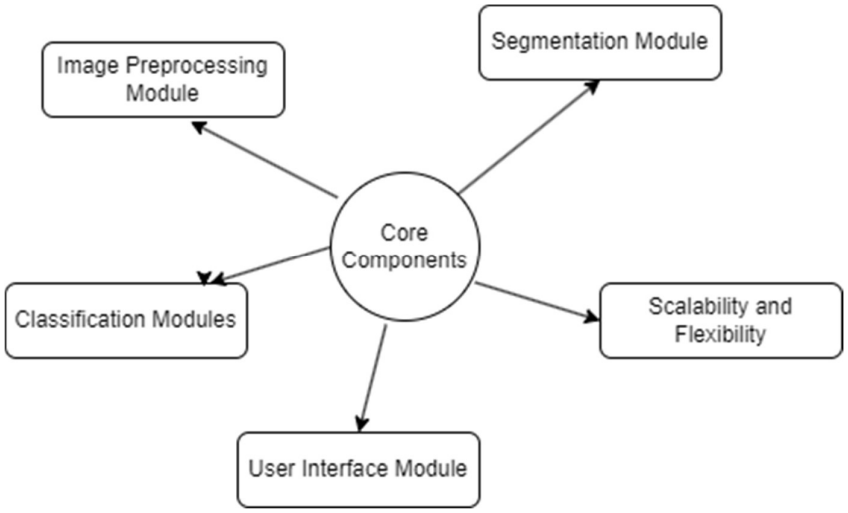


Figure 3.2 Core Components

1. Image Pre-processing Module: At the core of the automated LCLU system is the image preprocessing module, which utilizes powerful python modules to perform the preprocessing with the large sized UAV images by breaking them into smaller chunks/packets. We'd used python's Open CV module for displaying the images, but later on it was not able to display large sized or tif images. So the currently image displaying and preprocessing module is GDAL and raestrio.

2. Segmentation Module: At the core of the system is the segmentation module, which employs OBIS techniques to partition images into meaningful objects. This module uses clustering algorithms such as K-Means and advanced segmentation techniques like watershed segmentation. By identifying and isolating distinct land cover features, the system facilitates more precise classification.

3. Classification Modules: The classification module leverages various machine learning algorithms, including Random Forest, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and deep learning models implemented in TensorFlow. These algorithms analyse the extracted features and classify land cover types with high accuracy. The system can handle both supervised and unsupervised classification, allowing it to adapt to different datasets and requirements.

4. User Interface Module: The system is accessible through a user-friendly interface developed using Tkinter, a Python GUI toolkit. This interface provides intuitive tools for uploading imagery, configuring analysis parameters, and visualizing results. It includes features such as user authentication, secure access controls, and real-time monitoring of classification processes, ensuring a smooth and secure user experience.

5. Scalability and Flexibility: One of the significant advantages of this system is its scalability. The SAMAR system is designed to handle large volumes of imagery and extensive datasets, making it suitable for organizations of varying sizes, from small research teams to large environmental agencies. The flexible architecture allows for easy updates and integration of new features, ensuring that the system can adapt to evolving technological advancements and user needs.

3.3 Technological Infrastructure

The SAMAR system is built on a robust technological foundation that ensures reliability, efficiency, and scalability. The primary technologies employed include:

- **Python**: Chosen for its versatility and extensive libraries, Python serves as the core programming language for developing the system's functionalities.
- **Tkinter**: A standard Python interface to the Tk GUI toolkit, Tkinter is used to develop the graphical user interface, providing a seamless and interactive user experience.
- **Machine Learning Algorithms**: These algorithms are integral to the feature extraction and classification modules, enabling the system to learn from vast amounts of data and improve its performance over time.
- **Object-Based Image Segmentation (OBIS):** Advanced segmentation techniques are utilized to partition imagery into meaningful objects, enhancing the accuracy of the classification.
- **Geographic Information Systems (GIS):** GIS technologies are employed for spatial data analysis and visualization, providing valuable insights into land cover and land use patterns.

By integrating these advanced technologies, the SAMAR system aims to revolutionize the LCLU classification landscape, providing a more efficient, accurate, and reliable means of analyzing and managing land use. This system not only reduces the manual effort and time required for LCLU analysis but also ensures consistent and unbiased results, ultimately contributing to a more effective and equitable approach to environmental monitoring and management.

3.4 Need of Automated Land Cover and Land Use

The need for land cover and land use (LCLU) classification arises from the difficulties and inefficiencies inherent in traditional LCLU methods. Today's systems rely solely on manual processes for data analysis and classification, and there are many problems that can be reduced by using automated systems.

Resource Intensiveness

One of the biggest challenges to LCLU's approach is its vast resources. Manual analysis and classification of land cover data is a difficult task that requires the expertise and significant investment of researchers and analysts. This process often requires careful analysis of satellite imagery, extensive geographic coverage, and reprocessing to ensure that the classification is accurate and consistent with the real world. In addition, processing large volumes of big data increases the burden on staff, as analysts must evaluate each section individually, which is especially important in areas with increasing numbers of people. This huge burden on human resources not only puts pressure on analysts, but also distracts them from other important aspects of environmental monitoring and management responsibilities.

Accuracy and Consistency Challenges

Despite their expertise and good intentions, human analysts are prone to biases and errors in classification. Terms included in the satellite imagery dictionary may produce different results depending on the judgment and experience of the observer. This problem is particularly evident in areas with fine or complex land cover features, where individual biases and different methods can influence the classification process. This inconsistency not only affects the reliability and accuracy of the distribution, but can also lead to a lack of trust in LCLU data among stakeholders and stakeholders.

Scalability Issues

Many methods of manually editing LCLU classification have become untenable. Traditional LCLU management methods are impractical and increase in difficulty and effectiveness as the process progresses. This is especially problematic for large environmental studies and projects that require large areas of land cover to be distributed across multiple regions. Manual processes cannot effectively scale to meet increasing data needs, creating a significant administrative and operational burden that slows down the classification process and introduces potential errors. Faced with these various challenges, the need to move to an automated LCLU delivery system arose..

The Imperative for Automation

In light of these multifaceted challenges, the imperative for transitioning to an automated LCLU classification system becomes abundantly clear. Automated systems can dramatically reduce the resource burden on analysts by streamlining the data analysis and classification processes through advanced algorithms and machine learning techniques. These systems can analyze large datasets quickly and accurately, ensuring comprehensive and consistent classifications without the extensive time and effort required in manual processes.

In conclusion, the adoption of automated LCLU classification systems is not merely a technological advancement but a necessary evolution in environmental monitoring and management. By addressing the inherent inefficiencies, accuracy challenges, consistency issues, and scalability problems of traditional methods, automated systems provide a more efficient, accurate, and reliable framework for LCLU analysis, ultimately enhancing the quality and effectiveness of environmental monitoring efforts.

3.5 Advantages of Land Cover and Land Use Systems

The advantages of automated Land Cover and Land Use (LCLU) classification systems are extensive, significantly enhancing the efficiency, accuracy, and scalability of environmental monitoring and management processes. These benefits are derived from the integration of advanced machine learning algorithms, remote sensing techniques, and modern data processing frameworks, which collectively empower automated systems to streamline operations, ensure precise classifications, and support large-scale environmental analysis.

Enhanced Efficiency and Productivity

One of the foremost advantages of automated LCLU systems is their ability to dramatically enhance efficiency and productivity throughout the classification lifecycle. By automating the

labor-intensive processes of data analysis and classification, these systems significantly reduce the time and effort required from analysts, enabling them to allocate their resources more effectively towards other critical environmental monitoring and management tasks. Automated systems can rapidly process large volumes of satellite imagery and other geospatial data, ensuring timely updates and comprehensive coverage of land use patterns, which is essential for effective environmental planning and decision-making.

Moreover, the automation of data processing eliminates the need for manual interpretation, which can be particularly time-consuming in the case of large and complex datasets. Advanced machine learning algorithms can analyze and classify land cover features with high accuracy, providing timely and actionable insights to support environmental conservation and management efforts. This efficiency gain not only enhances the overall productivity of analysts but also enriches the quality of environmental data by ensuring consistency and reducing the potential for human error.

Improved Accuracy and Consistency

Automated LCLU systems offer a significant improvement in the accuracy and consistency of land cover classifications. Traditional manual classification methods are prone to human error and subjective biases, leading to variability in results. Automated systems, on the other hand, apply standardized algorithms and criteria uniformly across all data, ensuring consistent and unbiased classifications. Machine learning algorithms, trained on extensive datasets, can recognize and classify complex land cover features with high precision, reducing the likelihood of misclassification and enhancing the reliability of the results.

Additionally, automated systems can continuously learn and adapt from new data, improving their accuracy over time. This continuous improvement capability ensures that the system remains up-to-date with evolving land use patterns and environmental conditions, providing more accurate and reliable classifications for long-term monitoring and analysis.

Scalability and Flexibility

One of the inherent strengths of automated LCLU systems is their scalability and flexibility to accommodate the evolving needs of environmental monitoring and management. Traditional manual methods struggle to scale effectively to meet the demands of large and complex datasets, leading to logistical challenges and inefficiencies. Automated systems, however, are designed to handle large volumes of data efficiently, making them suitable for extensive environmental studies and projects that require comprehensive land cover analysis across vast geographic areas.

Moreover, automated systems are highly adaptable to different environmental contexts and requirements. They can be customized and configured to suit specific project needs, allowing for the integration of various data sources and the application of specialized classification algorithms. This flexibility ensures that automated systems can be tailored to address the unique challenges and objectives of different environmental monitoring projects, providing a versatile and scalable solution for diverse applications.

Technological Advancements and Innovation

Finally, automated LCLU systems drive technological advancements and innovation in the field of environmental monitoring. By leveraging cutting-edge technologies such as machine

learning, remote sensing, and geospatial data analysis, these systems push the boundaries of traditional classification methodologies and enable new and innovative approaches to land cover analysis. The continuous development and integration of advanced algorithms and techniques ensure that automated systems remain at the forefront of technological innovation, enhancing their capabilities and functionalities over time.

Moreover, the adoption of automated LCLU systems fosters interdisciplinary collaboration and knowledge exchange among environmental scientists, technologists, and researchers. This collaborative ecosystem promotes a culture of innovation and experimentation, driving forward-thinking initiatives and research endeavors aimed at further improving the accuracy, efficiency, and applicability of land cover classifications. By serving as a platform for technological innovation, automated LCLU systems contribute to the continuous advancement of environmental monitoring practices, ultimately supporting more effective and sustainable environmental management.

In summary, automated LCLU systems offer a multitude of advantages that transcend the limitations of traditional classification methods. From enhancing efficiency and productivity to improving accuracy and consistency, ensuring robust security and data integrity, and fostering technological innovation, automated systems represent a transformative force in the field of environmental monitoring and management. These systems empower analysts and institutions to conduct comprehensive, accurate, and reliable land cover classifications, ultimately enhancing the quality and effectiveness of environmental monitoring efforts and supporting informed decision-making for sustainable environmental management.

Chapter 4: System Design

System design is the process of designing the elements of a system such as the architecture, modules, and components, the different interfaces of those components, and the data that goes through that system.

4.1 Architecture of the Automated LCLU System

The architecture of the Automated Land Cover and Land Use (LCLU) classification system is designed to handle large sized images, perform complex image processing tasks by breaking them into smaller chunks/packets, and deliver accurate classification results. The system leverages modern technologies such as machine learning, GIS, and remote sensing to provide a robust and scalable solution.

4.1.1 High-level System Components

The system is composed of several key components that work together to achieve the desired functionality. The image selection and displaying layer. This image is then preprocessed to remove noise, correct geometric distortions, and enhance image quality, ensuring that the data is suitable for analysis and classification.

The Image Segmentation Module applies advanced segmentation algorithms, such as Object-Based Image Segmentation (OBIS), to divide the images into meaningful segments or objects. This module also performs feature extraction, where features such as texture, shape, color, and spectral properties are extracted from the segmented images. These features are crucial for accurate classification.

The Classification Module utilizes various machine learning models, including Random Forest, Support Vector Machines (SVM), and Convolutional Neural Networks (CNN), to classify the segmented images into different land cover and land use categories. The models are trained using labeled datasets and validated to ensure accuracy and reliability.

The GIS Integration Layer enables the integration of classified data into a Geographic Information System (GIS) for spatial analysis. This layer allows for the visualization of classified data on maps and supports further spatial queries and analyses.

The User Interface provides a user-friendly dashboard that allows users to interact with the system, view classification results, and generate reports. The dashboard includes various visualization tools such as maps, charts, and graphs.

4.1.2 Interaction between Components

Data flows from the ingestion layer to the preprocessing module where it is cleaned and prepared for segmentation. Preprocessed data is then fed into the image segmentation module, which generates segmented objects and extracts features. These features are passed to the classification module where machine learning models classify the segments into different LCLU categories. The classified data is integrated into the GIS layer for spatial analysis and visualization.

4.2 Functional Requirements

The functional requirements of the Automated LCLU system define the essential capabilities and features that the system must possess to achieve its objectives.

The system must be able to ingest data from various sources including satellites, UAVs, and aerial photography, supporting multiple data formats such as raster and vector. It should have image correction and enhancement capabilities, including noise reduction and geometric correction features. The image segmentation module should implement advanced segmentation algorithms like OBIS and extract features such as color, texture, shape, and spectral indices from the segmented objects.

The classification module should support multiple machine learning algorithms, including training, validation, and testing of classification models. The system should ensure high accuracy and precision in classification results. The GIS integration layer should enable spatial analysis and visualization, supporting spatial queries and map-based analysis.

The user interface should be user-friendly, with intuitive navigation and visualization tools, providing comprehensive reporting and export functionalities.

4.2.1 Image Segmentation Module

The last of them is the Image Segmentation Module which is aimed at concerntration of images into meaningful segments/objects as well as extraction of the features for classification. It uses Object-Based Image Segmentation (OBIS) to divide images into region groups because of spectral and spatial similarities. Sophisticated techniques used include the edge detection, region growing, clustering, and others. From the segmented objects, features including color, texture, shape and the spectral indices are extracted from the segments and accumulate them into a-feature vector for use in classification..

4.2.2 Classification Module

The Classification Module uses the analysed data and performs classification of segmented objects by using machine learning techniques to categorise them as different land cover and/or land use types. Different algorithms like Random Forests, SVMs, and CNNs are used. Models are built with labeled data, and this data is tested to check the probability. The module divides all data into training and testing data sets and employs a cross-validation to check all the models under consideration and to avoid overfitting. Parameters of the model are optimized to get maximum accuracy. The module generates classified maps and reports; the unit connects the results of the classification with GIS for additional analysis and representation.

4.3 Non-functional Requirements

The non-functional requirements are statements of the emphatic attributes of the total system that covers performance, scalability, security and usability among others. The system has to be capable of quickly processing and analyzing data and use efficient algorithms and coding for big data to work well. It should be developed to be horizontally and vertically scalable so as to cater for higher data volumes and more users; leveraging on cloud solutions for scalability of storage and computing.

Normally, since the data being transmitted is customer related, strict measures have to be taken to protect the information so that there is no compromise to data integrity and confidentiality, data has to be transmitted encrypted and stored in the same manner. It is critical to make the user interface of the system as intuitive and as easily navigable as possible with clear

instructions and help sources provided to the users. The system's dependability must be established through validation and verification processes; the system should employ graceful failure mechanisms in the occurrence of system failures. Accessibility is also important with the system aimed to be easily maintainable and upgradable in future, constructed from modules to enable system components to be developed and deployed independently of one and other.

Thus, the Automated LCLU System seeks to meet these functional and non-functional requirements when accomplishing its purpose of offering the best method of land cover and use classification.

Chapter 5: SAMAR – **S**PACE BASED **A**NALYSIS FOR **M**ONITORING OF **A**GRO-**R**ESOURCES

Multi-purpose software called for land use and land classification is called SAMAR or Space Based Analysis for Monitoring of Agro-Resources, which is based on Python. It employs up-to-date satellite imagery analysis method, which is useful in evaluating the agricultural resources.

5.1 Introduction to SAMAR

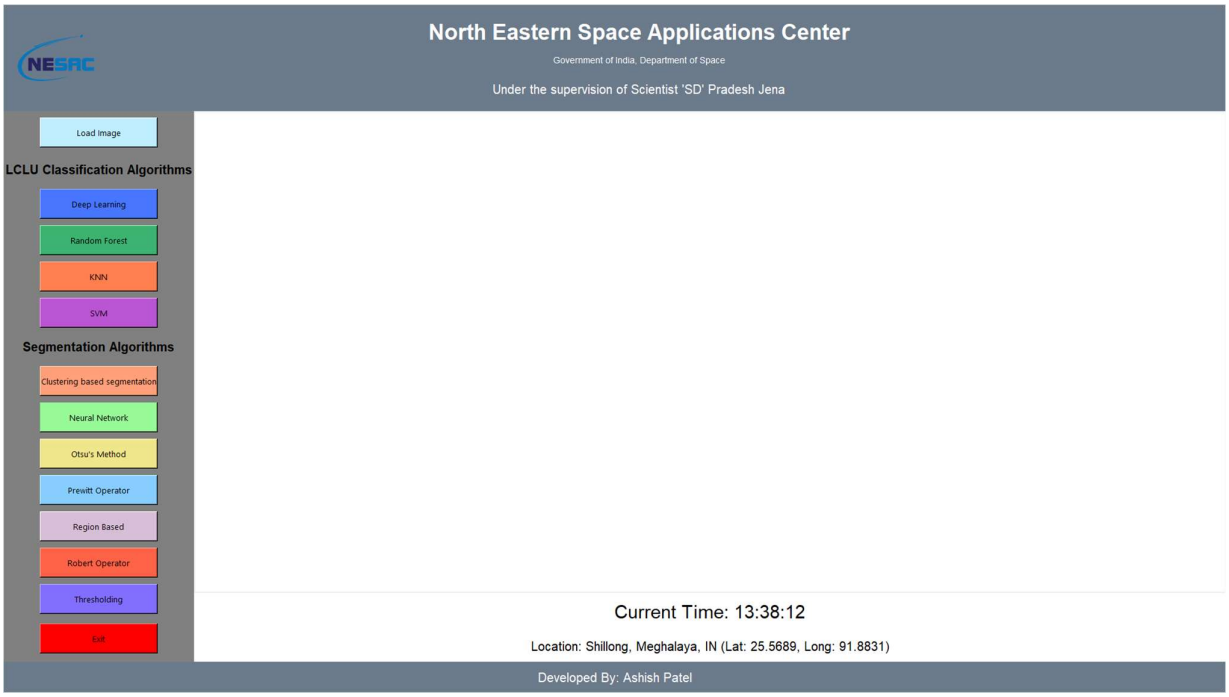


Figure 5.1 Homescreen of SAMAR

SAMAR is a Python-based software using Tkinter as the graphical user interface, which forms a versatile software for land use and classification analysis. It works through the need to take UAV imagery data and ground control points as data inputs for it to function. The software divides the captured image by the UAV into a number of small packets that can easily be handled for classification of land coverages that involve the use of sophisticated image processing in addition to the application of artificial neural networks. Such classifications have often involved the farming lands, forests, cities, rivers, roads, water bodies, and the wastelands.

The approach of the study adopted by SAMAR is the application of the supervised learning technique and the GCPs are then partitioned between training dataset and testing dataset at 70:30. The division of the area into subareas permits obtaining classification models that will allow the correct classification of the land's coverage in the UAV images. On the image data it uses supervised feature set for other classification algorithm such as Random Forest, KNN, Deep Learning, and SVM while for unsupervised classification it uses K-Means.

In this way, SAMAR contributes to the systematical agricultural monitoring and resource management, by offering exact spatial data on the changes of the land cover. Specifically, through the identification and mapping of different classes of land cover, SAMAR is useful in decision making processes concerning solving problems in agricultural, environmental as well as the management of the land.

Thus, SAMAR is a novel contribution to the set of spatial analysis tools, as it enables users effectively apply UAV imagery and GPC data to improve agricultural monitoring and resource assessment.

Objectives and Goals

As a Python-coded software with Tkinter as the GUI, SAMAR's objectives and goals focus on the promotion of spatial monitoring of agro-resources. SAMAR aims to achieve several key objectives and goals: SAMAR aims to achieve several key objectives and goals:

Firstly, SAMAR aims at improving the agricultural monitoring through the use of UAV imagery and use of GCPs. The major goal of the software is to differentiate or categorize the various types of land cover including agricultural land, forest land, urban built up, water bodies like rivers and lakes, roads and tracks, water bodies, and barren land. Due to the proper identification as well as mapping of these LC types, SAMAR enables effective monitoring and management of agro-resources.

Second, SAMAR to contribute to the improvement of sustainable Agriculture and resources. The software thus enhances the agricultural planning, water usage, as well as environmental protection through offering accurate geographical data referring to the latter aspects at the present time. This goal is relevant to the proper distribution of resources and the enhancement of sustainable development.

Another major goal of SAMAR is to equip the decision-makers with useful geo-information from the spatial analysis. Hence, the software provides accurate maps and classification of agro-resources that aids the decision-making of researchers, policy makers, and land managers. This capability includes informed policy and strategy formulation in regard to the development of agriculture and the social responsibility towards the environment.

On the technological level, SAMAR provides methods such as machine learning methods (supervised – namely, Random Forest for classification; unsupervised – SVM, Deep learning, KNN, and K-Means methods). These technologies make it possible for SAMAR to process and analyze the UAV imagery at scale and with the highest reliability irrespective to geographic and climatic entrenchment.

Additionally, SAMAR is used for future scientific and innovation development in the fields of remote sensing and agriculture. It serves for new methodological and algorithmic developments and helps to improve the methods of spatial data analysis. Thus, this aspect of SAMAR is to support and advance the advancement and development of innovative ways in the agricultural monitoring and spatial analysis.

Overall, SAMAR aims to enhance spatial analysis capabilities for monitoring agro-resources, support sustainable agricultural practices, empower decision-makers with actionable insights, integrate advanced technologies, and foster innovation in remote sensing and agricultural monitoring. These objectives collectively contribute to SAMAR's role in advancing agricultural development, environmental stewardship, and data-driven decision-making processes.

5.2 Technological Framework

An integrated technological foundation helps Samar, the Space Based Analysis for Monitoring of Agro-Resources, to include superior attributes such as geospatial as well as machine learning based analysis along with ease of use. In the following sub-sections, the focus is on the critical enablers that support and facilitate the accomplishment of SAMAR's objectives efficiently.

5.2.1 Geospatial Data Handling

Overall, the efficient functioning of the required functionalities greatly depends on the capacity of SAMAR in dealing successfully with geospatial data. In detail, the general functionality of SAMAR is based on the usage of the Geospatial Data Abstraction Library (GDAL) together with the Python access (GDAL). Thus, GDAL is a basic module of SAMAR, which allows working with different raster and vector GSI data sets without problems. This comprises changing of UAV (Unmanned Aerial Vehicle) imagery data in formats like GeoTIFF and the management of ground control points (GCPs) data which is commonly in CVS format.

Thanks to completely implemented capabilities GDAL SAMAR can effectively and efficiently read and write into geospatial data also process it. DTE offers core functions like data transformation, projection management, and format conversion that enable the integration of disparate data sets. The incorporation of GDAL into the SAMAR structure allows for the direct calls of these functions from Python space, which has improved the software's flexibility.

Compared to other libraries, for instance, OpenCV, commonly used for image processing tasks, GDAL stands out by being the best for working with big geo-referenced datasets, which can be characteristic of UAV imagery. OpenCV could, however, have issues with the memory and the processing power required to handle large UAV images especially since the library is designed to perform computer vision tasks as compared to geospatial data processing while GDAL was designed with spatial data formats in mind and consequently, it has features aimed at optimizing the handling of large raster datasets presumably without any loss in efficiency. Machine Learning and Data Analytics

Among all the capabilities, it is supervising learning which is most efficient in SAMAR and includes the Random Forest classifiers, Deep learning, SVM, and KNN. Such classifiers are usually built on labeled datasets of Ground Control Points (GCPs) that have a very significant role in achieving a good result in the classification of land cover. Explicitly, the algorithms, such as Random Forest, Support Vector Machine (SVM), Deep Learning, K-Nearest Neighbours (KNN) is preferable to employ in this case, because they belong to the scikit-learn package, enable to manage huge amount of data and prevent overfitting, as well as allowees providing accurate predictions. SAMAR corrects UAV imagery and related GCP data so as to determine the spectral characteristics of the area selected for examination and subsequent training of the Random Forest, SVM neural network, Deep Learning, KNN model. This approach makes it easy for SAMAR to classify the types of land cover including agricultural land, forest land, urban area, river, roads, water and barren land in a very accurate manner.

Unsupervised Learning with K-Means Clustering:

In addition to supervised learning, SAMAR employs unsupervised learning techniques to extract insights from UAV imagery without relying on predefined labels. K-Means clustering is a pivotal algorithm used for segmenting the UAV imagery into distinct clusters based on spectral similarities. This segmentation helps SAMAR identify and delineate different land cover classes within the imagery. By grouping pixels into clusters based on their spectral

characteristics, SAMAR can effectively map out regions of interest and analyze spatial patterns across the landscape. This capability is particularly useful for applications where detailed segmentation and understanding of land cover dynamics are essential, such as environmental monitoring and urban planning.

Integration and Scalability:

SAMAR's integration of scikit-learn ensures seamless implementation of machine learning algorithms within its workflow. The library's robustness and scalability enable SAMAR to handle large-scale datasets typical in remote sensing applications, making it suitable for analyzing high-resolution UAV imagery captured over extensive geographic areas. The use of Python not only facilitates the implementation of complex machine learning pipelines but also supports the integration of additional data processing and visualization tools, enhancing SAMAR's versatility and usability for diverse analytical tasks.

5.2.2 User Interface and Interactivity

SAMAR prioritizes user experience through its intuitive and functional user interface (UI), developed using Tkinter, Python's standard GUI toolkit. This choice of Tkinter ensures a seamless integration of SAMAR's advanced geospatial analysis capabilities with an accessible and user-friendly interface.

Functionalities and Features:

Tkinter enables SAMAR to offer a comprehensive range of functionalities designed to streamline spatial analysis tasks. Users can easily upload UAV imagery, select and configure parameters for image processing and analysis, and visualize results in a clear and interactive manner. The UI provides intuitive controls such as buttons for image upload, radio buttons for parameter selection (like the choice of ground control points), and progress bars to monitor the processing status.

Image Upload and Processing:

The UI's image upload feature allows users to select GeoTIFF-format UAV images directly from their local storage. Upon selection, SAMAR utilizes Tkinter's file dialog capabilities to seamlessly integrate the chosen image into its processing pipeline. This integration is crucial for handling large-scale imagery efficiently, as users can initiate tasks such as supervised classification or unsupervised clustering directly from the UI.

Parameter Configuration:

SAMAR's UI offers parameter configuration options that enhance flexibility in analysis. For supervised learning tasks, users can toggle between uploading Ground Control Points (GCP) or opting for unsupervised methods like K-Means clustering. This flexibility empowers users to tailor SAMAR's algorithms to their specific analytical needs, ensuring precise and relevant outcomes.

Result Visualization:

The UI plays a pivotal role in visualizing SAMAR's outputs effectively. Upon completion of image processing tasks, the UI displays classified land cover maps directly within the

application window. Users can interact with these visual outputs, zooming in/out and exploring detailed classifications such as agricultural areas, forests, urban zones, rivers, roads, water bodies, and barren lands. Tkinter's integration allows SAMAR to incorporate matplotlib for dynamic plotting, ensuring that users can interpret and analyze spatial data comprehensively.

Usability and Accessibility:

By leveraging Tkinter, SAMAR enhances usability for both technical users and domain experts involved in agricultural resource management and environmental monitoring. The UI's straightforward navigation and clear feedback mechanisms simplify complex geospatial analyses, making advanced remote sensing techniques accessible to a broader audience. This accessibility is critical for stakeholders seeking efficient solutions for land use planning, environmental impact assessment, and sustainable resource management.

Future Enhancements:

Future iterations of SAMAR's UI could explore enhancements such as 3D visualization capabilities for terrain analysis, integration with cloud-based data storage for scalability, and real-time data streaming for dynamic monitoring applications. These advancements would further enhance SAMAR's usability and expand its utility across diverse domains requiring sophisticated geospatial analysis tools.

In summary, SAMAR's UI development with Tkinter underscores its commitment to usability, functionality, and accessibility in geospatial analysis. By providing a robust interface for interacting with advanced machine learning and remote sensing algorithms, SAMAR empowers users to derive meaningful insights and make informed decisions regarding agro-resource management and environmental monitoring.

5.2.3 Framework Flexibility and Customization

SAMAR's technological framework is meticulously crafted to prioritize flexibility and customization, essential for accommodating diverse research needs and operational scenarios in the monitoring of agro-resources. This adaptability is rooted in several key design principles and features that empower users to tailor SAMAR to their specific requirements.

Modular Architecture:

At the core of SAMAR's design is its modular architecture, which facilitates seamless integration of additional algorithms, data sources, and functionalities. This modular approach ensures that SAMAR can incorporate new advancements in geospatial technology and machine learning methodologies without necessitating extensive reengineering. Each module within SAMAR is designed to operate independently yet integrate cohesively within the overarching framework, promoting scalability and extensibility.

Algorithm Integration:

SAMAR supports the integration of a wide array of algorithms tailored for geospatial analysis and machine learning tasks. This includes supervised learning algorithms like Random Forest classifiers, which excel in classifying land cover based on labeled ground control point (GCP) datasets. Additionally, SAMAR integrates unsupervised learning algorithms such as K-Means clustering, enhancing its capability to segment UAV imagery into distinct land cover classes without predefined labels. The framework's flexibility allows researchers and practitioners to

experiment with different algorithms, optimizing performance based on specific application requirements.

Data Source Flexibility:

SAMAR is designed to seamlessly handle diverse geospatial data sources, including UAV imagery stored in formats like GeoTIFF and ground control points (GCPs) provided in CSV files. This flexibility ensures that SAMAR can ingest, preprocess, and analyze data from various sources, accommodating differences in spatial resolution, spectral bands, and geographic extents. By supporting multiple data formats and protocols, SAMAR enhances interoperability and accessibility across different research domains and operational environments.

User-Specific Customization:

A pivotal aspect of SAMAR's framework is its ability to accommodate user-specific requirements and preferences. Through configurable parameters and user interface (UI) options, SAMAR allows users to tailor analytical workflows and visualization outputs according to their unique needs. For instance, users can adjust processing parameters such as image segmentation thresholds, classification thresholds, and validation metrics to align with specific research objectives or application contexts. This customization capability empowers users to derive actionable insights and make informed decisions based on comprehensive spatial data analysis.

Scalability and Future Readiness:

By embracing a flexible and customizable framework, SAMAR is well-positioned to evolve alongside advancements in geospatial technology, machine learning methodologies, and user demands. Future enhancements may include the integration of cloud computing resources for enhanced scalability, real-time data streaming capabilities for dynamic monitoring applications, and advanced visualization techniques for interactive spatial analysis. These strategic advancements ensure that SAMAR remains adaptive and future-ready, supporting ongoing innovation and addressing emerging challenges in agro-resource monitoring and environmental management.

In conclusion, SAMAR's framework flexibility and customization capabilities underscore its commitment to facilitating advanced geospatial analysis and machine learning in the monitoring of agro-resources. By enabling seamless integration of algorithms, accommodating diverse data sources, and supporting user-specific customization, SAMAR empowers researchers, practitioners, and stakeholders to leverage spatial data effectively for sustainable land use planning, environmental stewardship, and agricultural productivity enhancement.

5.3 Data Sources and Integration

SAMAR – Space Based Analysis for Monitoring of Agro-Resources relies on a variety of data sources to perform accurate and comprehensive land use and land cover classification. The integration of these diverse datasets is crucial to ensure robust and reliable analysis.

UAV Imagery

Unmanned Aerial Vehicle (UAV) imagery forms a primary data source for SAMAR. UAVs provide high-resolution, timely images of the earth's surface, which are essential for detailed spatial analysis. These images capture fine-grained details, making them ideal for distinguishing between different land cover types such as agricultural fields, forests, urban

areas, and bodies of water. The UAV imagery is processed and segmented into smaller packets to handle large datasets efficiently and ensure that the system can manage and analyze them without performance degradation.

Ground Control Points (GCPs)

Ground Control Points (GCPs) are vital for ensuring the accuracy of UAV imagery. In SAMAR, GCPs are distributed in a 30-70 ratio for training and testing. These points are precise locations on the earth's surface with known geographical coordinates, used to calibrate and validate the spatial accuracy of UAV images. By incorporating GCPs, SAMAR enhances the reliability of its classification results, ensuring that the mapped outputs accurately reflect real-world locations.

Pre-Marked Data from Kaggle

To train and validate the machine learning models within SAMAR, a significant portion of the data is sourced from Kaggle. This dataset includes images that are pre-marked with various land cover classifications such as forest, agricultural land, barren land, water bodies, and urban areas. The pre-labeled data provides a robust foundation for training supervised learning algorithms, enabling SAMAR to develop accurate classification models. Utilizing Kaggle's pre-marked data ensures that the models are exposed to a wide variety of examples, improving their generalizability and performance.

Sentinel Satellite Images from Google Earth

In addition to UAV imagery and GCPs, SAMAR integrates Sentinel satellite images obtained from Google Earth. Sentinel satellites, part of the European Space Agency's Copernicus program, offer multispectral images with various resolutions suitable for environmental monitoring. These images cover larger geographical areas compared to UAVs and provide valuable information on regional and global scales. By incorporating Sentinel images, SAMAR can perform large-scale land cover classification and monitor changes over time, supporting comprehensive agro-resource management.

Data Integration and Processing

Integrating these diverse data sources into a cohesive system is a complex task that SAMAR handles efficiently. The Geospatial Data Abstraction Library (GDAL) and its Python bindings (PyGDAL) play a crucial role in this integration process. GDAL allows SAMAR to read, write, and process different raster and vector geospatial data formats. It ensures that data from UAVs, GCPs, Kaggle, and Sentinel satellites can be seamlessly integrated, manipulated, and analyzed within a single platform.

To manage and process these large datasets, SAMAR employs various data preprocessing techniques. This includes image segmentation, normalization, and transformation to ensure consistency and compatibility across different data sources. The preprocessing steps also involve aligning the spatial resolution and coordinate systems of UAV and Sentinel images, ensuring that they can be accurately overlaid and compared.

By leveraging a combination of high-resolution UAV imagery, precise GCPs, pre-marked datasets from Kaggle, and extensive Sentinel satellite images, SAMAR creates a robust framework for land use and land cover classification. The seamless integration of these diverse data sources enhances the accuracy and reliability of the analysis, enabling SAMAR to provide valuable insights for monitoring and managing agro-resources effectively.

Chapter 6: Methodologies and Algorithms

SAMAR employs a variety of advanced segmentation methods and machine learning algorithms to process UAV images and classify land cover. These methodologies form the backbone of SAMAR's analytical capabilities, enabling accurate and efficient extraction of meaningful insights from geospatial data. The diverse and robust methods implemented in SAMAR are essential for tackling the complex challenges associated with land use and land cover classification. Each technique is selected and optimized based on its suitability for specific types of data and analysis goals, ensuring that the system can deliver high-precision results across various scenarios. Below is an in-depth look at the key methodologies and algorithms that drive SAMAR's performance:

6.1 Unsupervised Learning Algorithms

When labelled data and Ground Control Points (GCPs) are not available, SAMAR relies on unsupervised learning algorithms to perform image segmentation. Unsupervised learning does not require labelled input data, making it ideal for scenarios where obtaining labelled data is challenging or impractical. This approach enables SAMAR to identify patterns and structures within the data autonomously. One of the primary unsupervised learning algorithms used in SAMAR for image segmentation is K-Means clustering, implemented in the `ClusteringBasedSegmentation.py` script.

6.1.1 K-Means Clustering

Implementation in SAMAR

K-Means clustering is an efficient algorithm that partitions an image into clusters based on pixel similarity. It begins by selecting a predetermined number of cluster centers (centroids) randomly. Each pixel in the image is then assigned to the nearest centroid based on a distance metric, typically the Euclidean distance. This assignment forms initial clusters, grouping pixels with similar spectral characteristics.

Once the pixels are assigned to clusters, the centroids are recalculated by taking the mean of all pixels within each cluster. This update step adjusts the centroid positions to better represent the cluster members' average location in the feature space. The assignment and update steps are repeated iteratively until the centroids stabilize, ensuring the clusters have minimal variance.

Advantages of K-Means Clustering in SAMAR

In SAMAR, K-Means clustering is particularly useful for segmenting UAV imagery into distinct land cover classes, such as forests, water bodies, and agricultural fields, without the need for predefined labels. This method leverages the spectral characteristics of the pixels, allowing the algorithm to autonomously identify and segment meaningful regions in the image. This process enables SAMAR to perform initial segmentation without manual intervention, making it suitable for data-scarce environments.

K-Means clustering is also computationally efficient and scalable, which is essential for processing large UAV images that cover extensive areas. This efficiency ensures that SAMAR can handle high-resolution imagery and provide rapid segmentation results, facilitating timely analysis and decision-making.

The segmented regions generated by K-Means can serve as a preprocessing step for further analysis, such as supervised classification once labeled data becomes available. Additionally,

the clusters can be visualized to provide a clear and intuitive representation of different land cover types, aiding in the interpretation and validation of the segmentation results.

6.2 Supervised Learning Algorithms

For land cover classification, SAMAR employs a variety of supervised learning algorithms that leverage labeled Ground Control Points (GCPs) to train models. These algorithms use the labeled data to learn patterns and make accurate predictions on unseen data. The primary supervised learning algorithms used in SAMAR include Random Forest classifiers, Support Vector Machines (SVMs), K-Nearest Neighbors (KNN), and deep learning models.

Random Forest Classifier: Detailed in `rf.py`, the Random Forest classifier is an ensemble learning method that combines multiple decision trees to improve classification accuracy and robustness. The classifier is trained on a dataset of GCPs, which are labelled with land cover classes such as forest, agricultural, barren, water, and urban. Features extracted from the UAV images are used to train the Random Forest model, which is then applied to classify the entire image. The ensemble approach of Random Forest helps in reducing overfitting and improves the generalizability of the model.

Support Vector Machines (SVM): Implemented in `svm.py`, SVMs are powerful classifiers that find the optimal hyperplane separating different classes in the feature space. The hyperplane is determined by maximizing the margin between the closest points of the different classes, known as support vectors. SVMs are particularly effective in high-dimensional spaces and are used in SAMAR to classify land cover types by training on labeled GCPs. The robustness of SVMs to overfitting and their ability to handle non-linear decision boundaries through kernel tricks make them suitable for complex classification tasks.

K-Nearest Neighbors (KNN): The KNN algorithm, found in `knn.py`, is a simple yet effective classification method. It classifies a new data point based on the majority class among its k-nearest neighbors in the feature space. KNN is a non-parametric algorithm, meaning it makes no assumptions about the data distribution. In SAMAR, KNN is used to classify land cover by identifying the closest GCPs in the feature space and assigning the most common class label among them. This algorithm is intuitive and works well with small to moderately sized datasets.

Deep Learning Models: SAMAR also leverages deep learning techniques for land cover classification, particularly through convolutional neural networks (CNNs) implemented in scripts such as `yolo_v8_train.py` and `gui_yoloV8.py`. Deep learning models, especially CNNs, are highly effective in processing and classifying high-dimensional image data. These models learn hierarchical features from the raw pixel values of UAV images, capturing complex patterns and spatial hierarchies that traditional machine learning algorithms might miss. By training on large labeled datasets, deep learning models in SAMAR can achieve high accuracy in identifying and classifying various land cover types.

The integration of these supervised learning algorithms in SAMAR enhances its capability to perform precise and reliable land cover classification. By leveraging labeled GCPs and advanced machine learning techniques, SAMAR ensures accurate extraction of land cover information from UAV images, supporting detailed and insightful spatial analysis for monitoring agro-resources.

6.2.1 Random Forest Classifier

The `rf.py` script in the SAMAR project implements a supervised machine learning approach for land cover classification using UAV images and Ground Control Points (GCPs). The

primary algorithm utilized is the Random Forest classifier, which is well-suited for handling complex and high-dimensional datasets like UAV imagery. This script provides a robust method for classifying land cover types such as forest, agricultural, barren, water, and urban areas by leveraging labeled data points and efficiently processing large images in chunks.

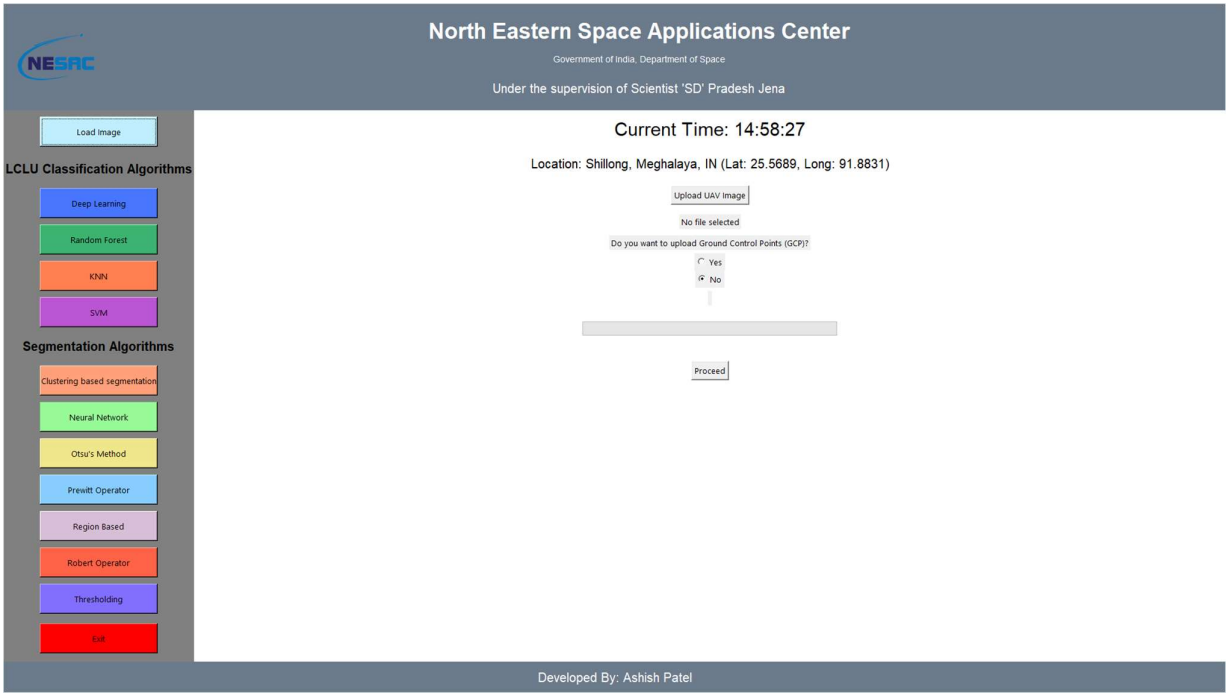


Figure 6.1 Random Forest in SAMAR

Algorithm for `rf.py`

1. **Load UAV Image:**
   - Use GDAL to load the UAV image file.
   - Extract image data and metadata (dimensions, number of bands).
2. **GCP Data Handling:**
   - If GCP data is provided, load the GCP CSV file using Pandas.
   - Extract features (coordinates) and labels (land cover types).
3. **Data Splitting:**
   - Split the GCP data into training and testing sets using `train_test_split`.
4. **Train Random Forest Classifier:**
   - Initialize the Random Forest classifier with specified parameters.
   - Train the classifier on the training data.
5. **Classify Image:**
   - Process the UAV image in chunks to manage memory usage.
   - Reshape each chunk and classify using the trained Random Forest model.
   - Assemble the predicted labels into a classified image.
6. **Display and Save Results:**
   - If the image is small enough, display the classified image using a color map.
   - Provide options to save the classified image in PNG or TIF formats.

Flowchart Diagram for Random Forest Algorithm:

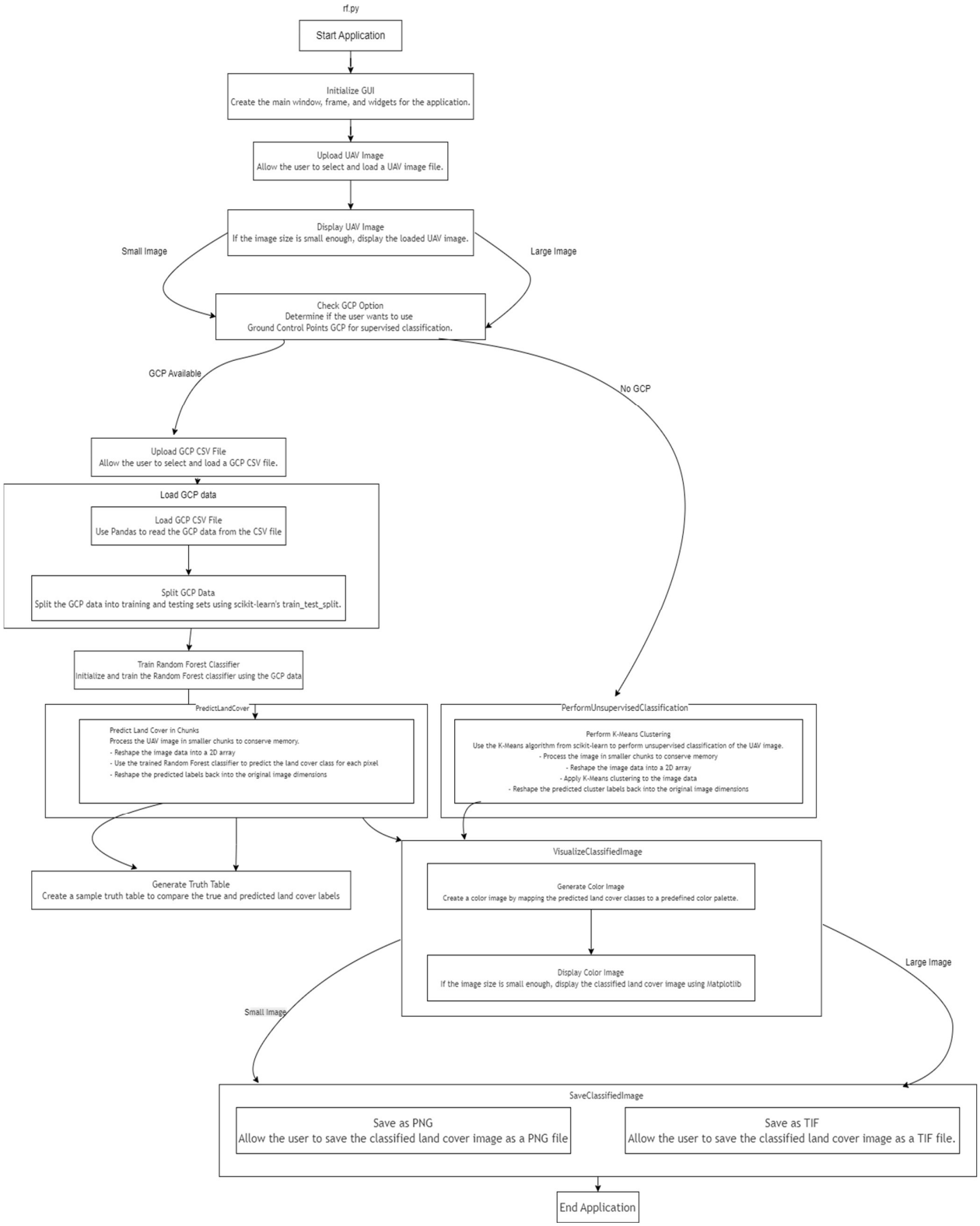


Figure 6.2 Random Forest Flowchart

6.2.2 Support Vector Machine

The `svm.py` script is designed to perform land cover classification on UAV (Unmanned Aerial Vehicle) images using Support Vector Machine (SVM) and K-means clustering algorithms. This script integrates with a Tkinter-based graphical user interface (GUI) to facilitate easy interaction with the classification process. The primary features include:

The `svm.py` script is designed to streamline the process of land cover classification using UAV (Unmanned Aerial Vehicle) images. One of its primary features is the ability to upload UAV images for classification. This functionality is integrated into a user-friendly Tkinter-based graphical user interface (GUI), allowing users to easily select and upload their UAV image files for processing. The script supports various file types, including TIF files, ensuring compatibility with a wide range of image formats commonly used in remote sensing and geographic information systems (GIS).

In addition to UAV image upload, the script offers the option to upload Ground Control Points (GCP) data in CSV format. This feature is particularly useful for supervised classification, as it allows the script to use the GCP data to train a Support Vector Machine (SVM) classifier. The inclusion of GCP data enhances the accuracy of the classification by providing labeled examples of different land cover types. Users can choose whether to upload GCP data based on their specific requirements and the availability of such data.

The `svm.py` script excels in performing land cover classification. When GCP data is available, the script trains an SVM classifier using the GCP data, and then uses this trained classifier to classify the UAV image. In the absence of GCP data, the script resorts to unsupervised classification using K-means clustering. This dual approach ensures that the script can handle both supervised and unsupervised classification scenarios, making it versatile and adaptable to different user needs and data availability.

Another notable feature of the script is its ability to display the results. The classified image, along with the original UAV image, can be displayed using matplotlib. This visual representation helps users to quickly assess the classification results and make informed decisions based on the visualized data. Additionally, the script provides options to save the classified images in both PNG and TIF formats. The ability to save classified images ensures that users can preserve and share their results for further analysis or reporting.

Finally, the `svm.py` script includes a feature to calculate the area covered by each land cover class. This is achieved by analyzing the classified image and computing the area for each class based on the image's geo-transform data. The calculated areas are then displayed to the user, providing valuable quantitative information about the distribution of different land cover types within the UAV image. This functionality is essential for applications such as environmental monitoring, urban planning, and agricultural management, where understanding the spatial distribution and extent of various land cover types is crucial.

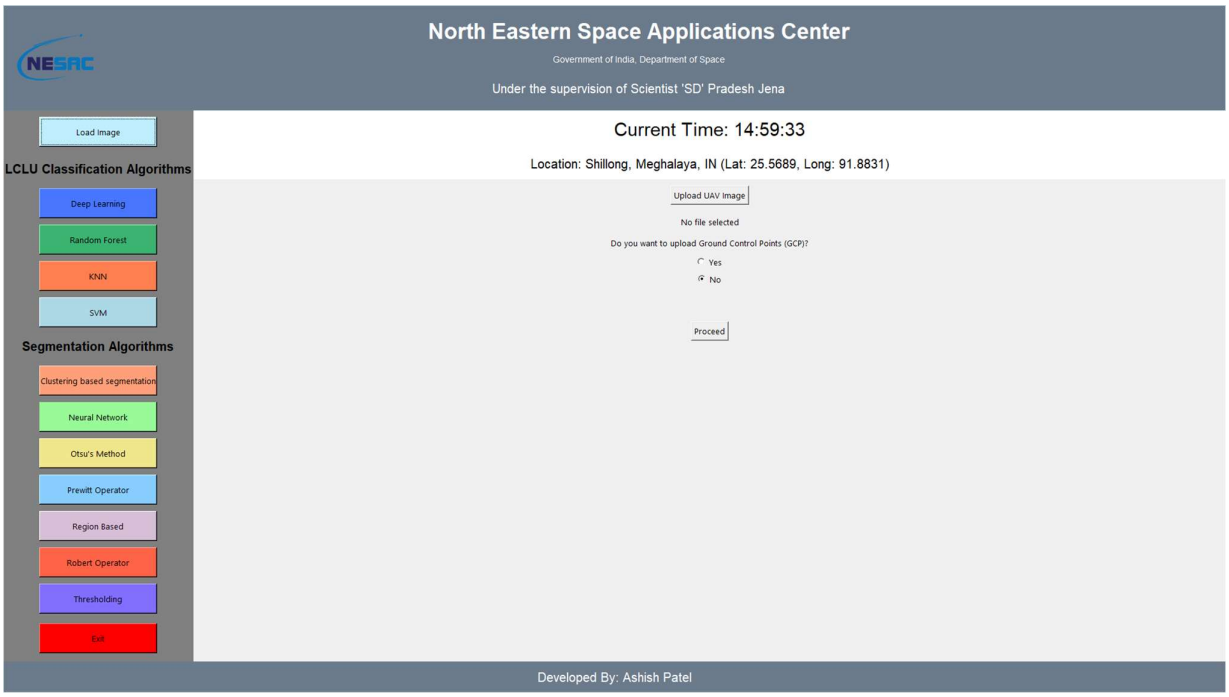


Figure 6.3 SVM in SAMAR

Algorithm for svm.py

The following pseudo code outlines the main steps performed by the `svm.py` script:

1. Initialize the Tkinter GUI
2. Define callback functions for GUI elements:
    a) select_image(): Open file dialog to select UAV image
    b) select_gcp(): Open file dialog to select GCP CSV file
    c) toggle_gcp_option(): Show/hide GCP file upload option
    d) display_image(image_data, title): Display image using matplotlib
    e) save_classified_image_png(): Save classified image as PNG
    f) save_classified_image_tif(): Save classified image as TIF
    g) calculate_area(classified_image, geo_transform): Calculate area of each class
    h) process_files(): Process the uploaded image and GCP data
3. Load UAV image and GCP data if available
4. If GCP data is available:
    a) Train SVM classifier using GCP data
    b) Classify UAV image using trained SVM
    c) Generate classified image and calculate accuracy using a truth table
5. If GCP data is not available:
    a) Perform unsupervised classification using K-means clustering
6. Generate classified image with color mapping for different land cover classes
7. Display classified image if it's not too large
8. Calculate and display area covered by each class
9. Save classified image in desired format

Flowchart Diagram for SVM Algorithm:

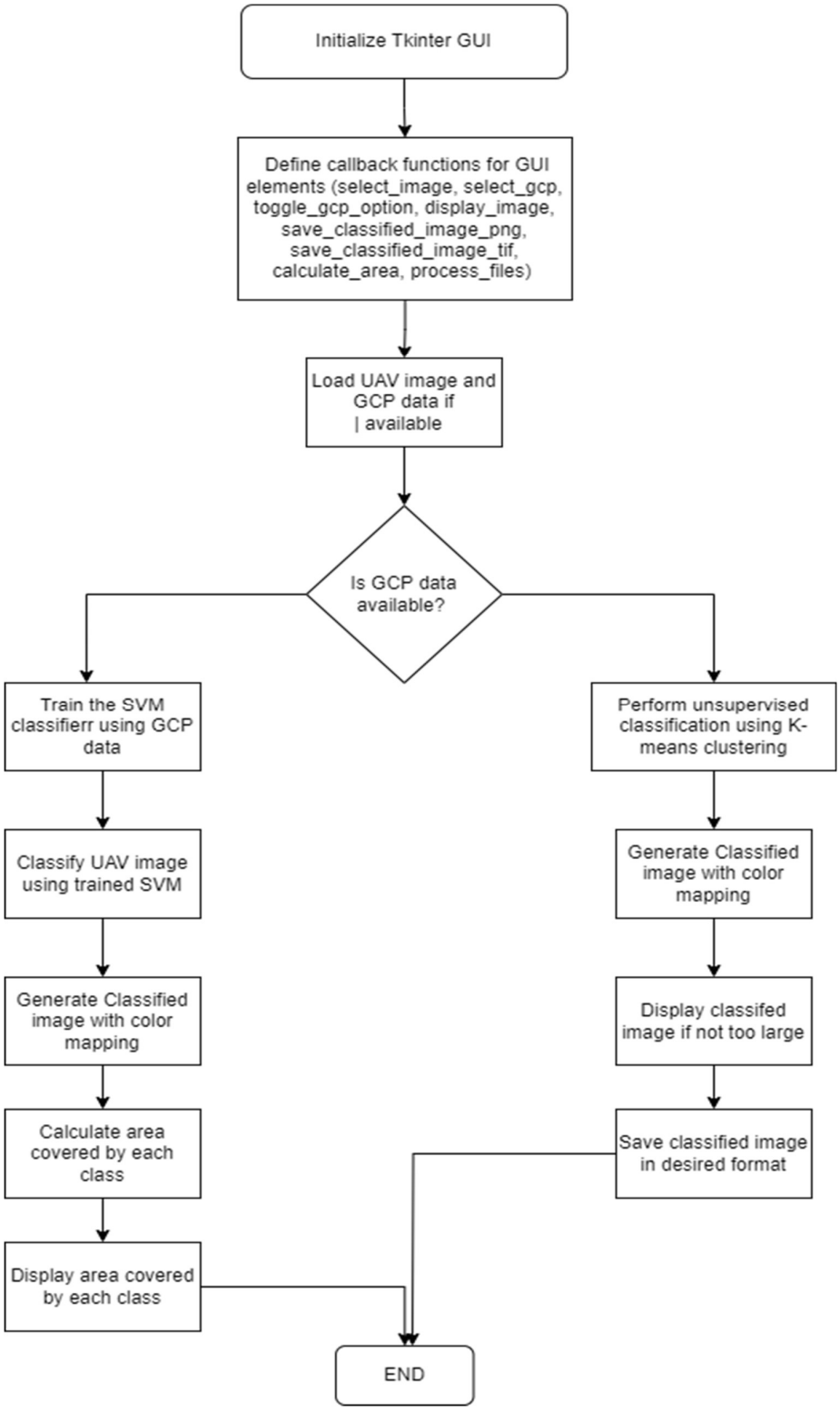


Figure 6.4 SVM Flowchart in SAMAR

6.2.3 K-Nearest Neighbor

The `knn.py` script is a robust tool designed to perform land cover classification using UAV (Unmanned Aerial Vehicle) images. It leverages the K-Nearest Neighbors (KNN) algorithm for supervised classification and K-means clustering for unsupervised classification, making it adaptable for various types of data and user needs. The script is integrated with a Tkinter-based graphical user interface (GUI), ensuring a user-friendly experience for uploading UAV images, optionally uploading Ground Control Points (GCP) data, processing the images, and visualizing the classification results. This script is particularly valuable for applications in environmental monitoring, urban planning, and agricultural management, where accurate land cover classification is crucial.

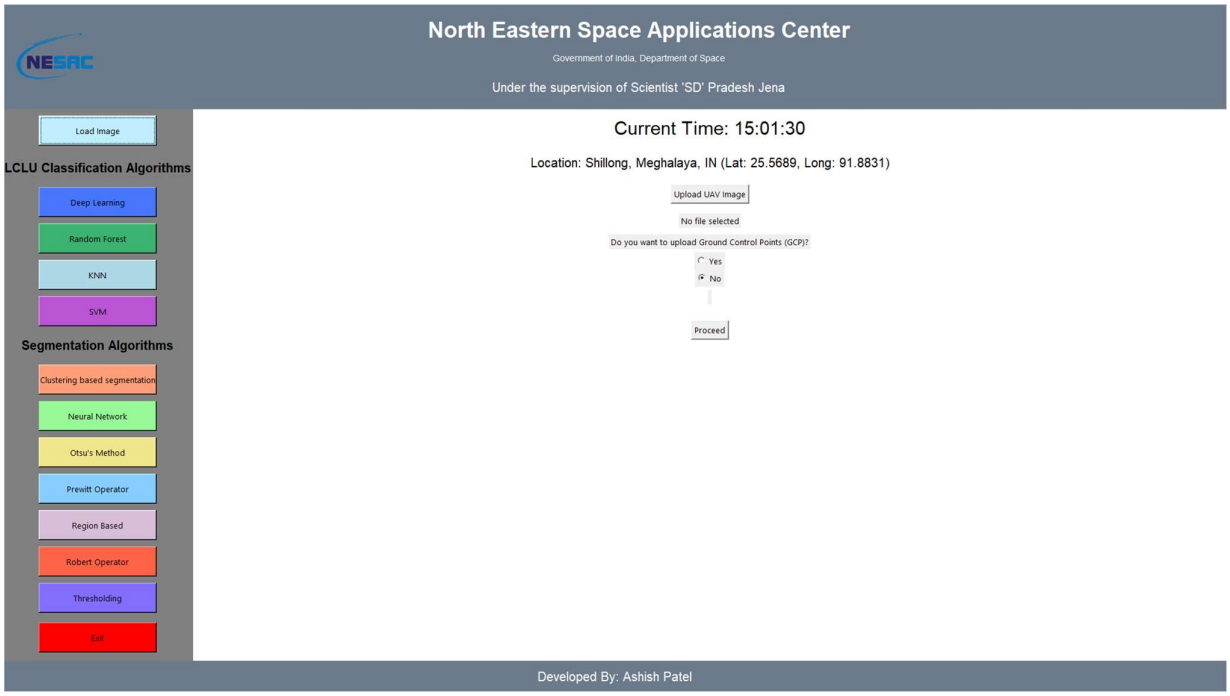


Figure 6.5 KNN in SAMAR

Algorithm for `knn.py`

1. **Initialize Tkinter Window**
   i. Create a root window using Tkinter.
   ii. Initialize state variables such as image_path, gcp_path, and progress_text.
2. **Setup User Interface**
   i. Create and configure UI components such as buttons for uploading images, GCP, and proceeding with processing.
   ii. Pack the buttons for user interaction.
3. **Select Image**
   i. Open a file dialog to select the UAV image.
   ii. Update the label to display the selected image path.
4. **Select GCP**
   i. Open a file dialog to select the GCP CSV file.
   ii. Update the label to display the selected GCP file path.
5. **Toggle GCP Option**
   i. Show or hide the GCP upload button based on the user's choice.
6. **Display Image**
   ii. Display the image using Matplotlib with a specified title.
7. **Save Classified Image as PNG**

i. Check if the classified image exists.
    a. If it exists, open a file dialog to select the save location.
    b. Save the image as a PNG file using Matplotlib.
    c. Display a success message.
ii. If it does not exist, display an error message.

8. Save Classified Image as TIF

i. Save Classified Image as TIF
    a. If it exists, open a file dialog to select the save location.
    b. Save the image as a TIF file using GDAL.
    c. Display a success message.
ii. If it does not exist, display an error message.

9. Process Files

i. Check if the UAV image is selected.
    - If not, display an error message and return.
ii. Load the UAV image using GDAL.
iii. Check if the image is large:
    a) If it is large, display a progress message and skip displaying the image.
    b) Otherwise, display the UAV image.
iv. Check if the GCP path exists:
    a. If it exists, load GCP data from the CSV file.
    b. Split the data into training and testing sets.
    c. Train the KNN classifier using the training data.
    d. Predict land cover classes for the image in chunks.
    e. Generate a truth table.
v. If the GCP path does not exist:
    a. Perform K-means clustering for unsupervised classification.
    b. Predict land cover classes for the image in chunks.
vi. Create a color image for visualization using predefined color maps.
vii. Check if the image is small enough:
    - If it is small, display the classified image using Matplotlib.
viii. Display a processing completion message.
ix. Show buttons for saving classified images.

10. Main Execution

i. Create the Tkinter root window.
ii. Initialize the KNNApp with the root window.
iii. Start the Tkinter main loop to run the application

Flowchart Diagram for KNN Algorithm

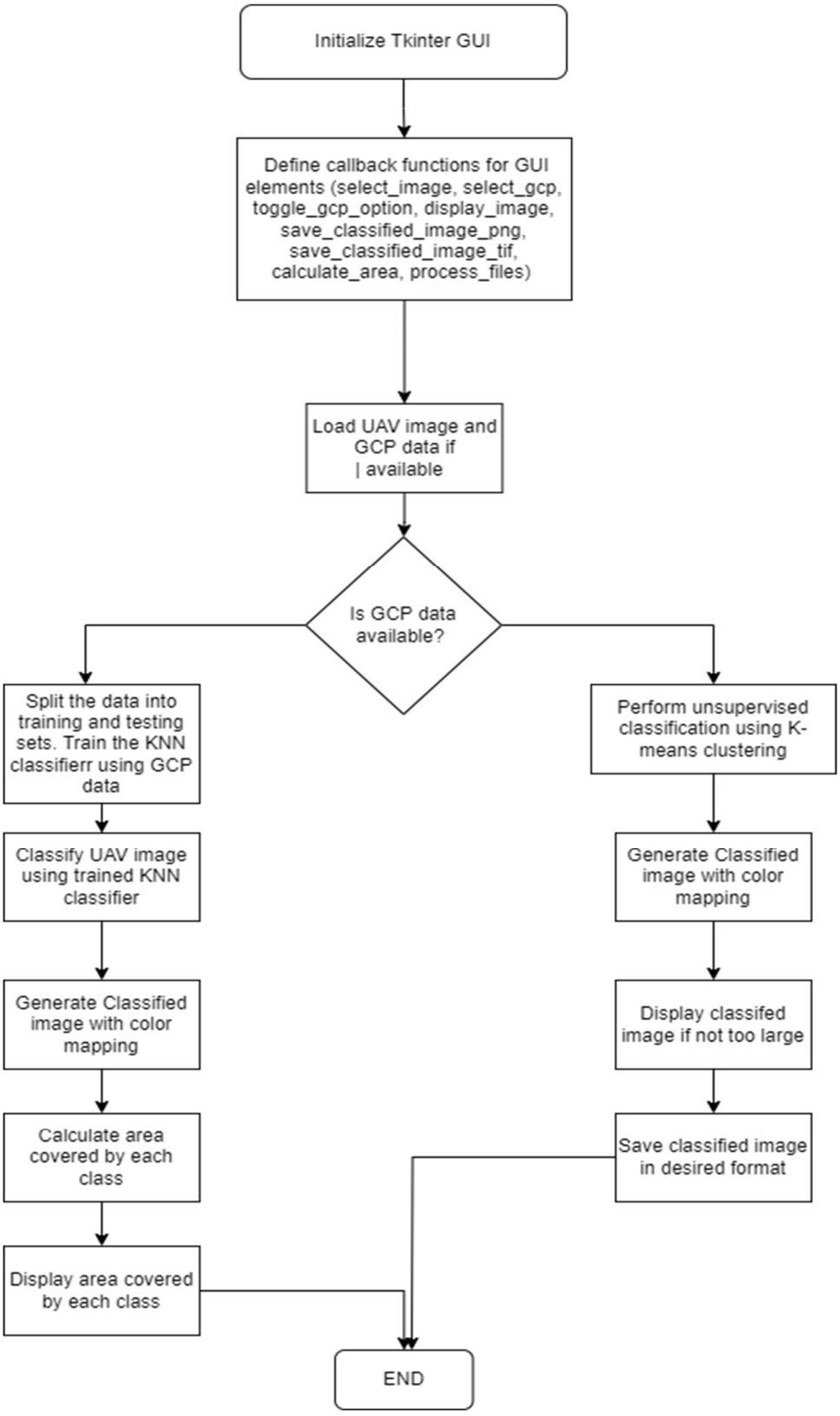


Figure 6.6 KNN Flowchart in SAMAR

6.2.4 Deep Learning

Deep learning is a subset of machine learning that involves the use of artificial neural networks with many layers to model and understand complex patterns in data. These networks, known as deep neural networks, are capable of learning representations of data with multiple levels of abstraction, making them particularly effective for tasks such as image recognition, natural language processing, and autonomous driving. Deep learning models, such as Convolutional Neural Networks (CNNs) for image processing and Recurrent Neural Networks (RNNs) for sequential data, have significantly advanced the state of the art in many areas of artificial intelligence.

One of the key advantages of deep learning is its ability to automatically extract features from raw data, reducing the need for manual feature engineering. By training on large datasets, deep neural networks can learn to recognize intricate patterns and make accurate predictions. The training process involves adjusting the weights of the network through a method called backpropagation, which minimizes the error between the predicted and actual outputs.

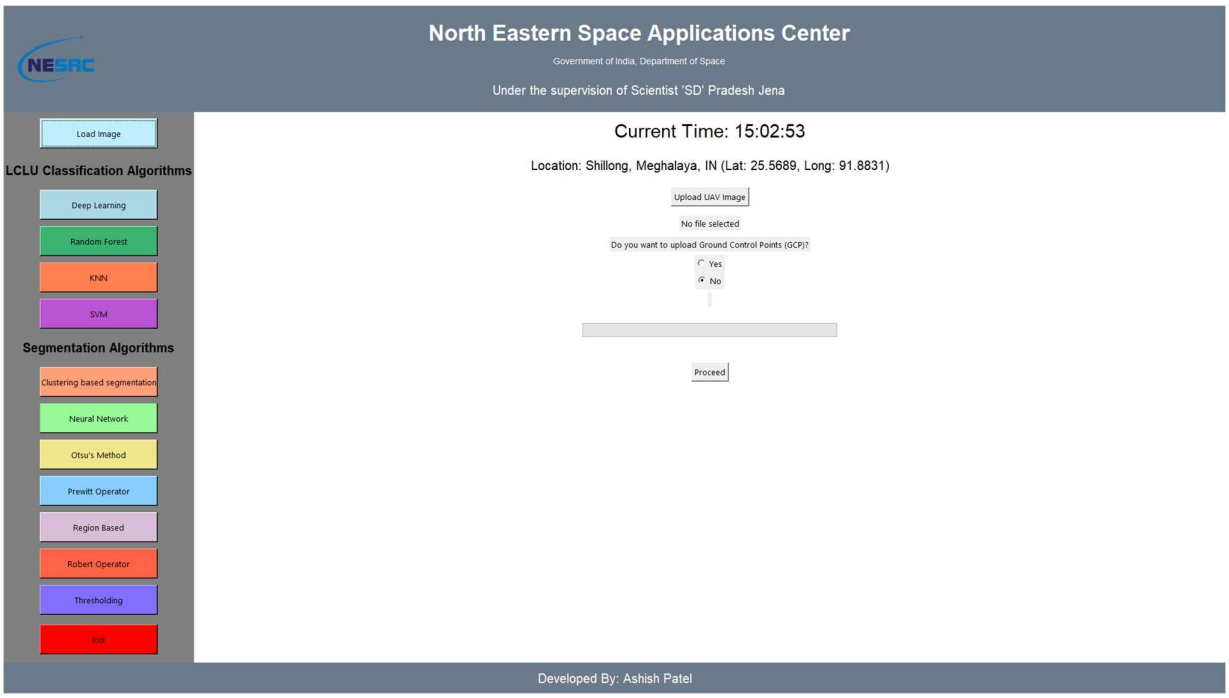


Figure 6.7 Deep Learning in SAMAR

Pseudo Code

The following pseudo code outlines the process of training a YOLO model and using it for image classification within a Tkinter GUI application.

Training the YOLO Model

1. Initialize YOLO model with pretrained weights:
   model = YOLO('yolov8n-seg.pt')
2. Train the model with custom dataset:
   model.train(data='config.yaml', epochs=70, imgsz=640, boxes=False)

YOLO GUI Application

1. **Initialize GUI application:**
    i. Create root window
    ii. Initialize YOLO model with trained weights
    iii. Read and initialize results counter
    iv. Setup UI components (buttons, labels, progress bar, etc.)
2. **Define function to read results counter:**
    i. Try to read counter from file
    ii. If file not found, initialize counter to
3. **Define function to write results counter:**
    i. Write the updated counter to file
4. **Define function to select image:**
    i. Open file dialog to select image
    ii. If image is selected, display it (or show message for TIFF images)
5. **Define function to proceed with prediction:**
    i. Check if image is selected
    ii. Update progress bar and label
    iii. If TIFF image, load with GDAL and prepare for YOLO prediction
    iv. Perform YOLO prediction
    v. Update progress bar and label
    vi. Update prediction results in text box
    vii. Enable save button
6. **Define function to display image:**
    i. Open and resize image
    ii. Update image label with the resized image
7. **Define function to display message:**
    i. Update image label with message
8. **Define function to save image:**
    i. Check if predicted image path is available
    ii. Open save file dialog
    iii. Save image with legend and details
9. **Define function to add legend and details to image:**
    i. Open predicted image
    ii. Draw legend and prediction details on image
    iii. Save the modified image
10. **Run the GUI application main loop**

Flowchart Diagram for Deep Learning Algorithm:

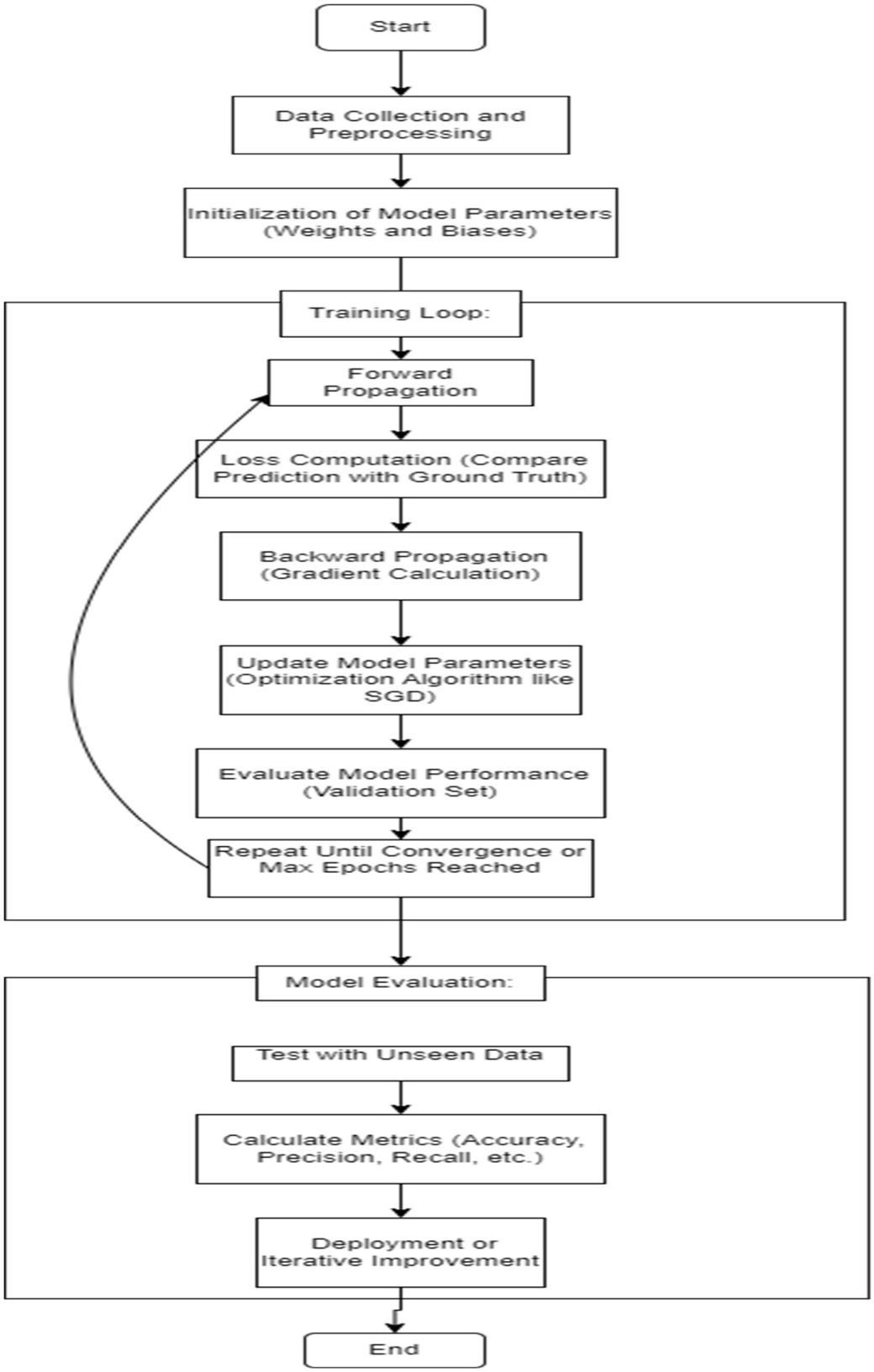


Figure 6.8 Deep Learning Flowchart

6.3 Segmentation Algorithms

Segmentation algorithms are a family of algorithms that group pixels in an image based on shared characteristics. These characteristics can include color, intensity, texture, or spatial location. The algorithms then assign labels to each pixel in the image.

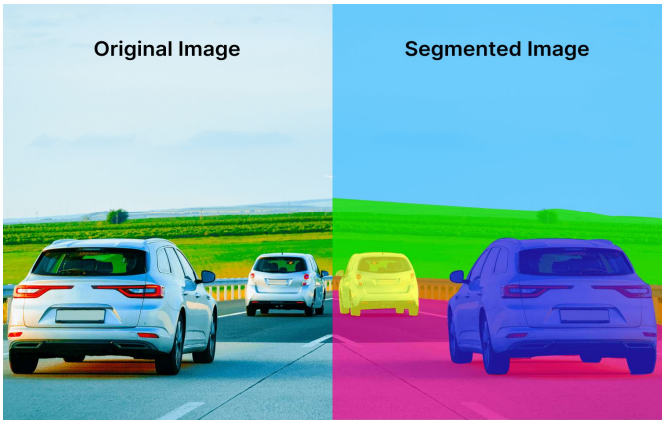


Figure 6.9 Image Segmentation

6.3.1 Clustering Based Segmentation

Clustering-based segmentation is a technique that divides an image into regions or segments by grouping their pixels by their similar characteristics. This method is particularly useful in identifying different objects or regions within an image. K-means clustering is one of the most commonly used algorithms for this purpose.

In K-means clustering, the aim is to partition the pixels into k clusters where each pixel belongs to the cluster with the nearest mean value. Here, we will discuss a Tkinter-based application for performing K-means clustering on an image, the algorithm behind it, and provide a flowchart to illustrate the process.

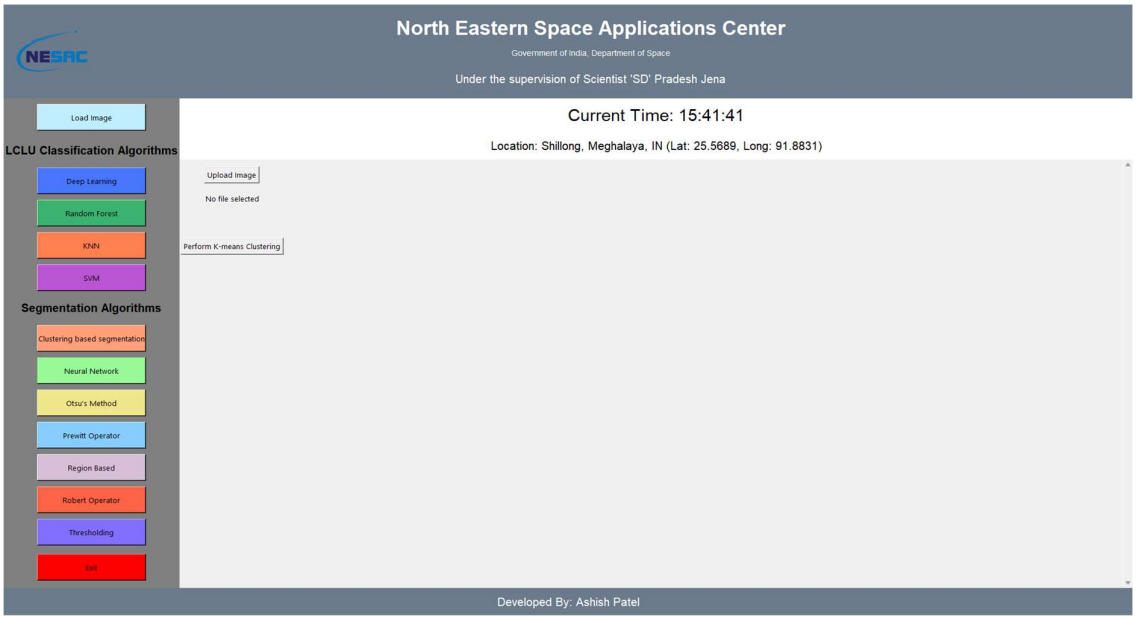


Figure 6.10 Clustering Based Segmentati

Algorithm for `ClusteringBasedSegmentation.py`

The algorithm for K-means clustering in the given application can be summarized as follows:

1. **Initialization**:
   - Initialize Tkinter window and state variables.
   - Load the selected image and display it in the GUI.
2. **Image Upload**:
   - Select the input image using a file dialog.
   - Load and display the image using PIL (Python Imaging Library).
3. **Perform K-means Clustering**:
   - Initialize cluster centers randomly.
   - Assign each pixel to the nearest cluster center.
   - Recalculate the cluster centers as the mean of the assigned pixels.
   - Repeat the assignment and update steps until the cluster centers converge.
4. **Display Results**:
   - Display the final cluster centers and the segmented image.

Flowchart Diagram for Clustering Based Segmentation

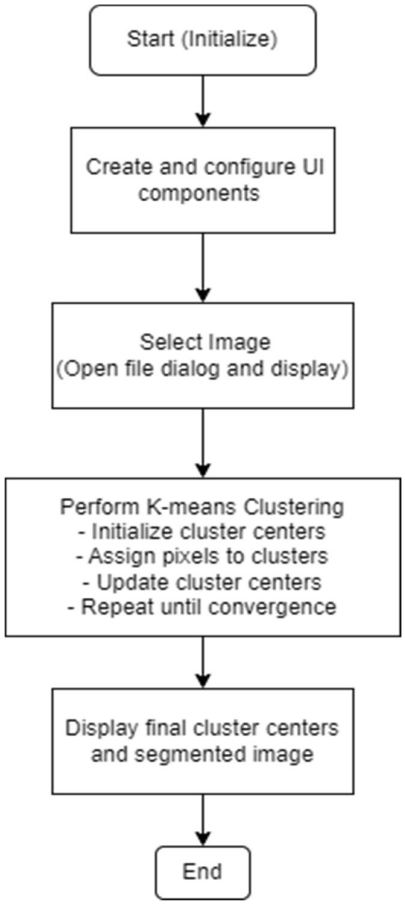


Figure 6.11 Clustering Segmentation Flow

6.3.2 Otsu's Thresholding Method of Image Segmentation

A common image segmentation method for converting a grayscale picture into a binary image is Otsu's thresholding. In an effort to minimize the intra-class variance of the black and white pixels, the method selects a threshold. When the image has a bimodal histogram—that is, two distinct peaks suggesting the foreground and background—this segmentation technique appears especially helpful.

Using a Tkinter-based GUI program, the accompanying Python code implements Otsu's thresholding segmentation. We will look over the application, its algorithm, and a process flowchart diagram below.

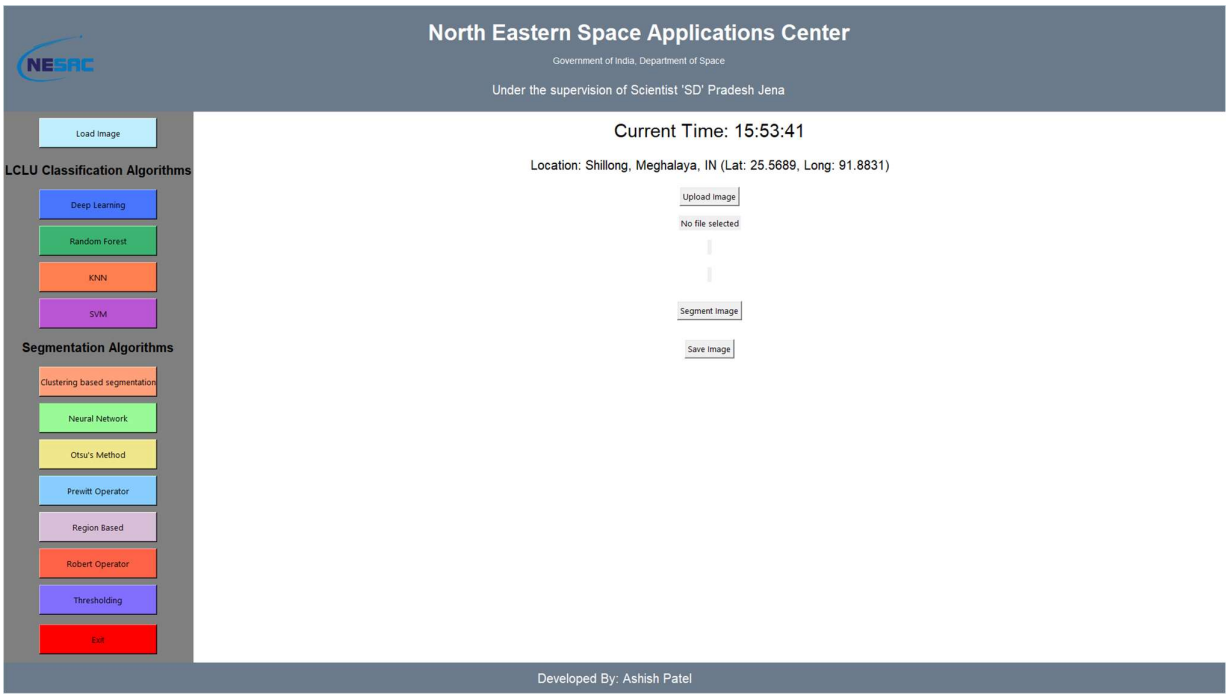


Figure 6.11 Otsu's Segmentation in SAMAR 1

Algorithm for `Otsus.py`

The algorithm for Otsu's thresholding segmentation in the given application can be summarized as follows:

1. **Initialization**:
   - Initialize Tkinter window and state variables.
   - Create and configure UI components.
2. **Image Upload**:
   - Select the input image using a file dialog.
   - Load and display the image using PIL (Python Imaging Library) and OpenCV.
3. **Segmentation**:
   - Convert the input image to grayscale.
   - Apply Otsu's thresholding to obtain the binary segmented image.
   - Display the segmented image in the GUI.
4. **Save Segmented Image**:
   - Allow the user to save the segmented image as a PNG file.

Flowchart Diagram for Otsu's Method

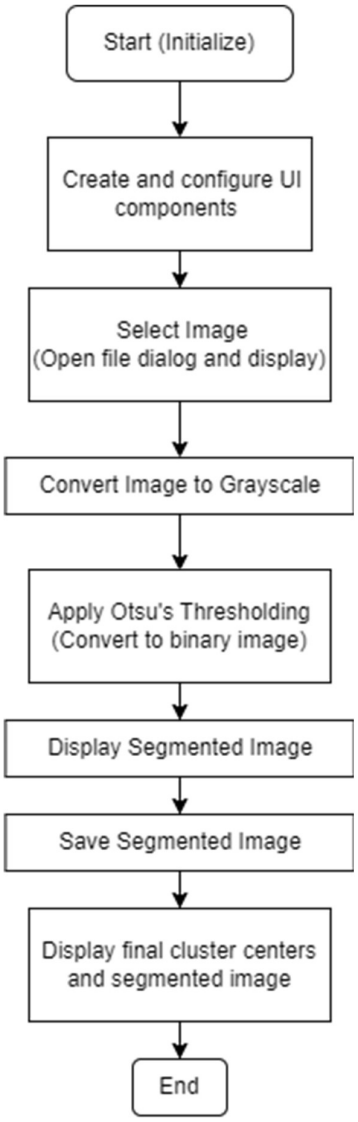


Figure 6.12 Otsu's Segmentatin Flowchart

6.3.3 Prewitt Edge Detection Application

The Prewitt edge detection algorithm is used to detect edges in an image by computing the gradient of the image intensity. The gradient is approximated using the Prewitt operator, which involves convolving the image with two 3x3 kernels that respond to vertical and horizontal changes in intensity.

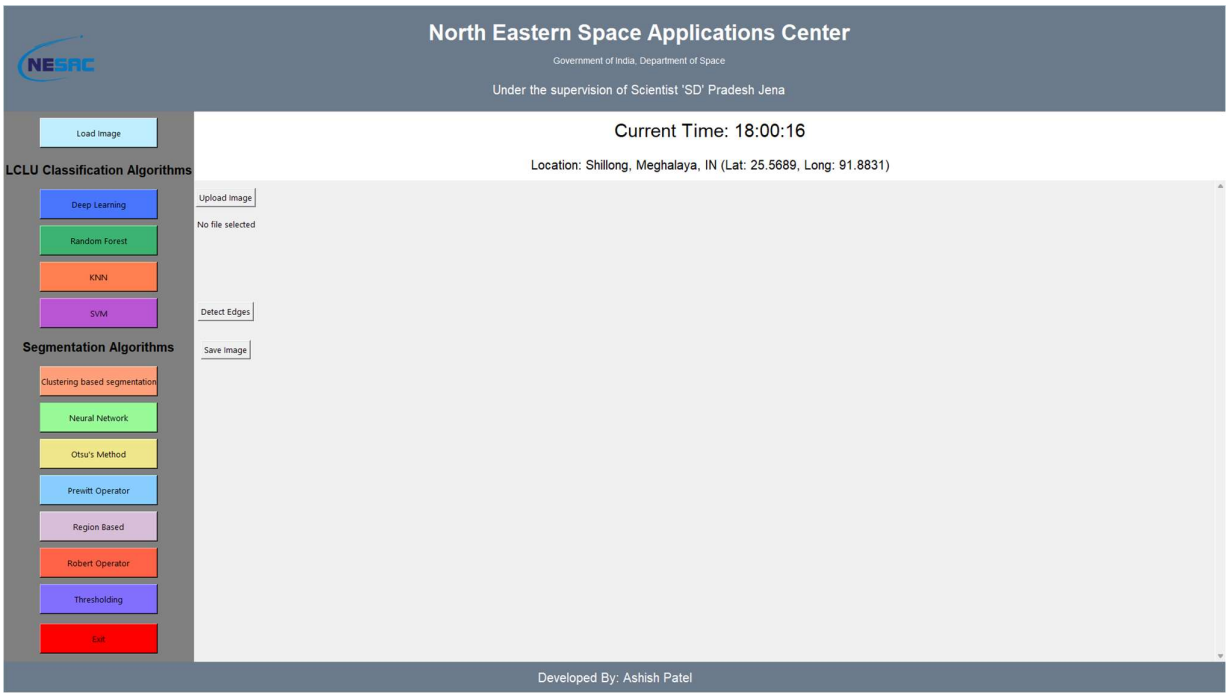


Figure 6.13 Prewitt Edge Detection

Algorithm for `prewitt.py`

- **Initialization**:
    - Set up the Tkinter window and UI components.
    - Define variables to store the input image and processed images.
- **Image Upload**:
    - Allow the user to select an image file using a file dialog.
    - Load the image using GDAL for various file formats.
    - Display the image if its resolution is below a certain threshold.
- **Edge Detection**:
    - Convert the input image to grayscale.
    - Apply the Prewitt operator by convolving the grayscale image with horizontal and vertical Prewitt kernels.
    - Compute the gradient magnitude from the horizontal and vertical gradients.
    - Threshold the gradient magnitude to create a binary edge-detected image.
    - Display the filtered and edge-detected images.
- **Save Image**:
    - Save the edge-detected image as a PNG file.

Flowchart diagram for Prewitt Edge Detection Algorithm

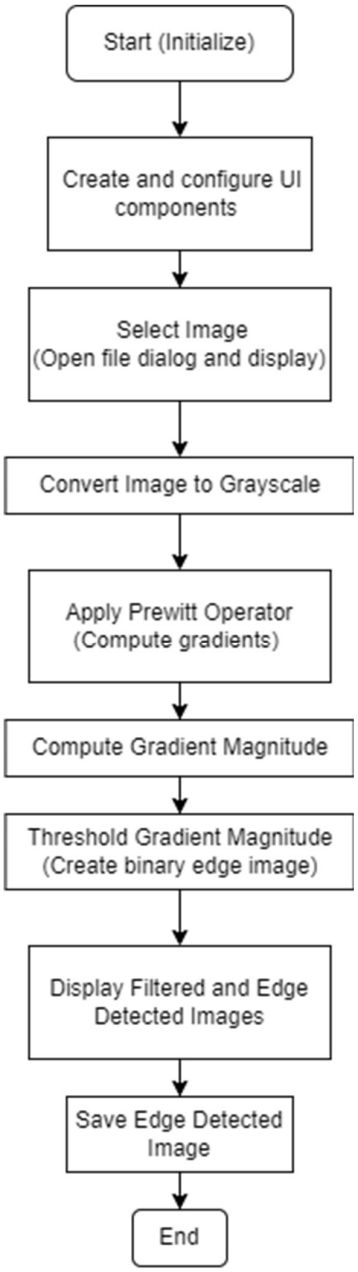


Figure 6.13 Prewitt Edge D Flowchart

6.3.4 Watershed Segmentation Application

The watershed segmentation algorithm is used to segment an image into regions based on the gradient of the image intensity. The gradient is computed using the Sobel operator, and markers are created to guide the segmentation process.

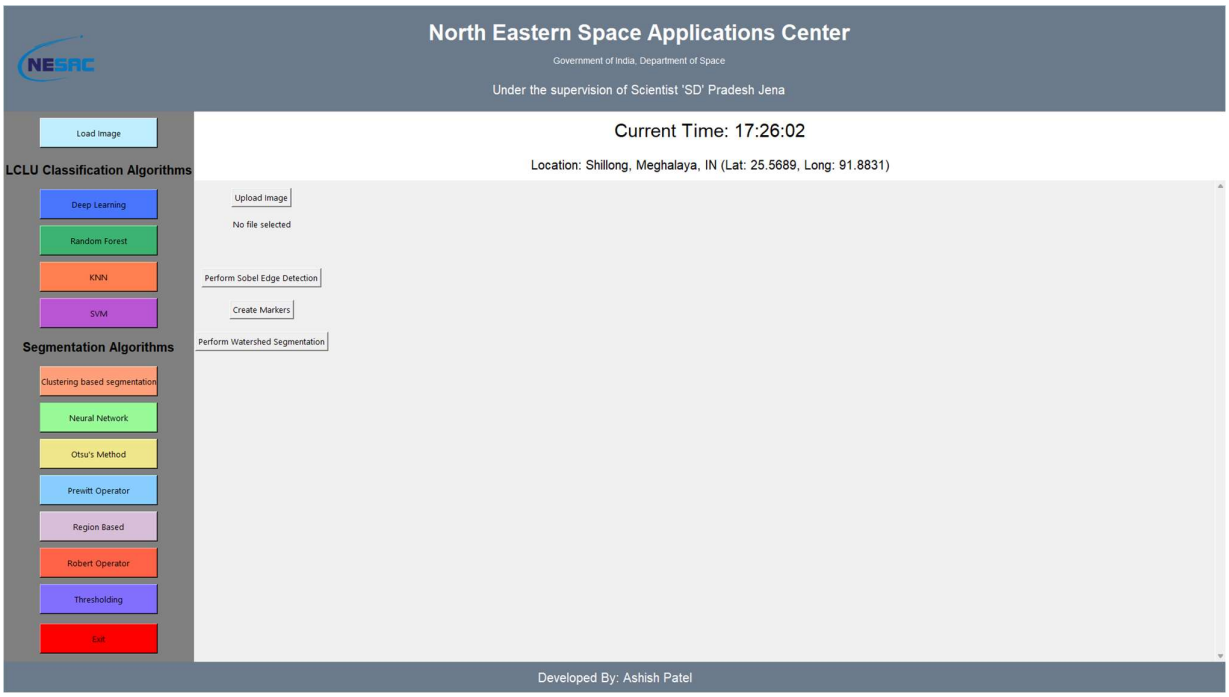


Figure 6.14 Watershed Segmentation

Algorithm for `regionbasedsegmentation.py`

1. **Initialization**:
    a. Set up the Tkinter window and UI components.
    b. Define variables to store the input image and processed images.
2. **Image Upload**:
    a. Allow the user to select an image file using a file dialog.
    b. Load the image using OpenCV and display it in the GUI.
3. **Edge Detection**:
    a. Convert the input image to grayscale.
    b. Apply the Sobel operator to compute the gradient (elevation map) of the grayscale image.
    c. Display the elevation map.
4. **Marker Creation**:
    a. Create markers based on the elevation map to guide the watershed segmentation.
    b. Display the markers.
5. **Watershed Segmentation**:
    a. Perform watershed segmentation using the elevation map and markers.
    b. Fill holes in the segmented regions and label the regions.
    c. Create an overlay of the labeled regions on the original image.
    d. Display the segmented image.

Flowchart diagram for watershed segmentation

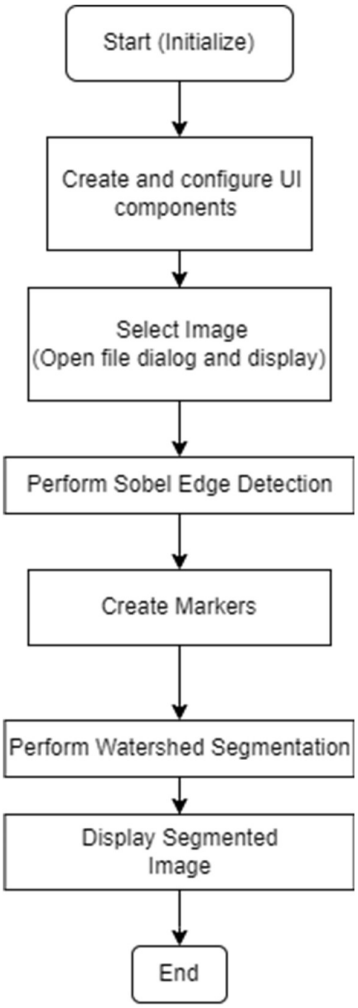


Figure 6.15 Watershed Segmentation Flow

6.3.5 Robert Edge Detection Application

The Robert operator is used to detect edges in an image by calculating the gradient of the image intensity at each pixel. The gradient is computed using two 2x2 convolution masks that approximate the derivatives of the image intensity.

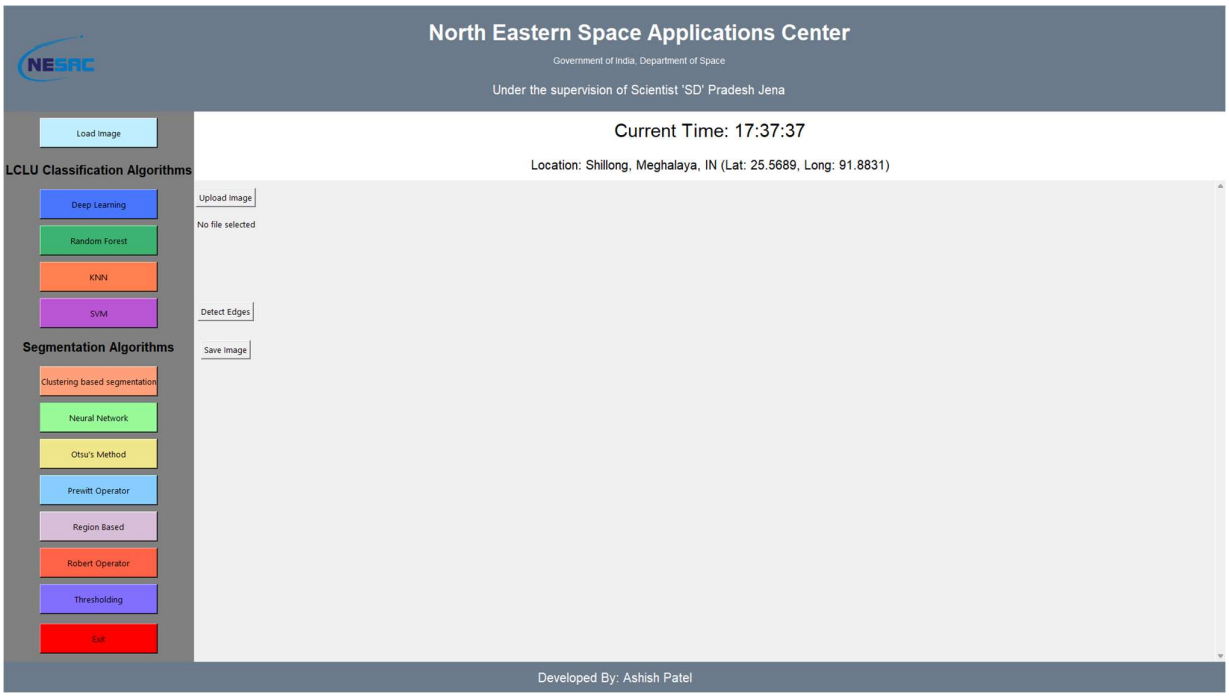


Figure 6.16 Robert Edge Detect in SAMAR

Algorithm for `Robert.py`

1. **Initialization**:
   - Set up the Tkinter window and UI components.
   - Define variables to store the input image and processed images.
2. **Image Upload**:
   - Allow the user to select an image file using a file dialog.
   - Load the image using GDAL and handle large images appropriately.
   - Display the image in the GUI if it is below a certain resolution threshold.
3. **Edge Detection**:
   - Convert the input image to grayscale.
   - Apply the Robert operator to compute the gradient of the grayscale image.
   - Threshold the gradient to obtain a binary edge map.
   - Display the edge-detected image and the intermediate filtered image using matplotlib.
4. **Save Image**:
   - Allow the user to save the edge-detected image to a file.

Flowchart Diagram for Robert Edge Detection Application

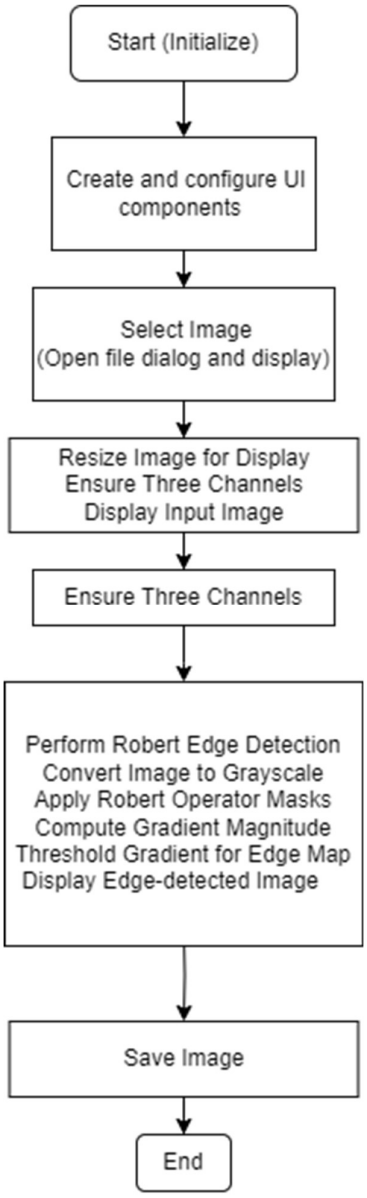


Figure 6.17 Robert Edge Detection Flowchart

6.3.6 Thresholding Segmentation

The thresholding segmentation is a Tkinter-based application that allows users to perform image thresholding. Users can upload an image, adjust the threshold value using a slider, process the image to apply the threshold, and save the processed image. The application includes a progress bar to indicate the processing progress, especially useful for large images.

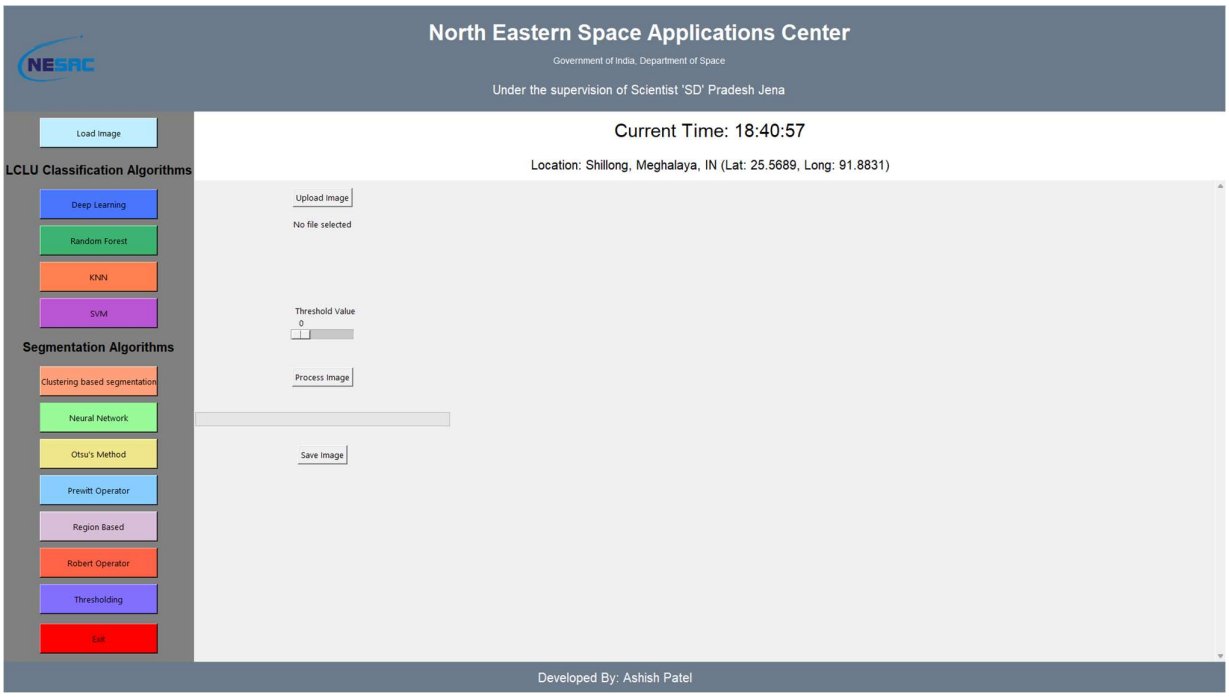


Figure 6.17 Thresholding Seg in SAMAR

Algorithm for `thresholding.py`

1. **Image Selection**:
    a. Users select an image using a file dialog.
    b. The image is loaded and displayed on the GUI.
2. **Threshold Value Adjustment**:
    a. Users adjust the threshold value using a slider.
3. **Image Processing**:
    a. The application processes the image by comparing each pixel value to the threshold value.
    b. If a pixel value is greater than the threshold, it is set to 255 (white).
    c. If a pixel value is less than or equal to the threshold, it is set to 0 (black).
    d. The progress bar updates as the image is processed.
4. **Displaying Results**:
    a. The processed (thresholded) image is displayed on the GUI.
    b. Optionally, the original and processed images can be displayed using Matplotlib.
5. **Saving the Processed Image**:
    a. Users can save the processed image using a file dialog.

Flowchart for Thresholding Segmentation

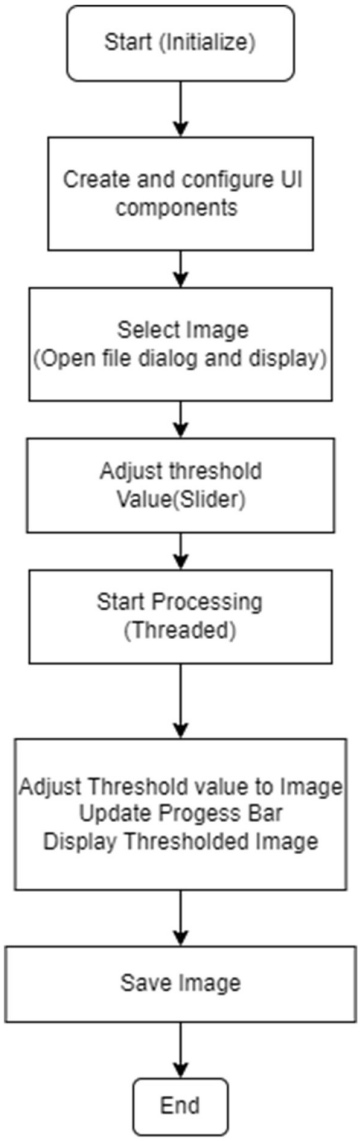


Figure 6.18 Thresholding Segmentation Flowchart

References

Gangadharan, A. (2020). *Geospatial Data Abstraction Library (GDAL) in Python and Anaconda.* Research Gate.

GDAL official Documentation. (n.d.). *GDAL*.

Kaul, H. (2012). *Land Use Land Cover Classification and Change Detection Using High Resolution Temporal Satellite Data.* Research Gate.

Otsu, N. (1979). *A Threshold Selection Method from Gray-Level Histograms.* IEEE.

Pedregosa, F. (2011). *Scikit-learn: Machine Learning in Python.* Journal of Machine Learning Research.

Qin, C.-Z. (2020). *GDAL/OGR and Geospatial Data IO Libraries.* Research Gate.

Sarker, I. H. (2021). *Machine Learning: Algorithms, Real-World Applications and Research Directions.* Springer.

Wasser, L. (2019). *EarthPy: A Python package that makes it easier to explore and plot raster and vector data using open source Python tools.* Journal of Open Source Software.