**Extracting Roads from Satellite Data for Effective Disaster Response**

**1. INTRODUCTION:**

Road extraction in urban areas has been an important task for generating geographic information systems (GIS). Especially in recent years, the rapid development of urban areas makes it urgent to provide up-to-date road maps. The timely road information is very useful for the decision-makers in urban planning, traffic management and car navigation fields, etc. Nowadays, we are experiencing an explosion in the amount of satellite image data, which provides us abundant data and also brings challenges to the road extraction task at the same time. The conventional road extraction methods by manual are time consuming and tedious, and cannot meet the increasing requirement for such tremendous data. Therefore, it has drawn considerable attention of many researchers on how to develop automatic road extraction systems. And much work has been done for this task. However, automatic extraction of urban roads from high resolution remote sensing imagery is still a challenging problem in digital photogrammetric and computer vision. The main reason is that the diverse road surfaces and the complex surrounding environments such as trees, vehicles and shadows induced by high buildings make the urban roads take on different textures and gray levels in images.

**1.1 Objective of the project:**

This paper focuses on automatic road extraction in urban areas from high resolution satellite images. We propose a new approach based on machine learning. First, many features reflecting road characteristics are extracted, which consist of the ratio of bright regions, the direction consistency of edges and local binary patterns. Then these features are input into a learning container, and AdaBoost is adopted to train classifiers and select most effective features. Finally, roads are detected with a sliding window by using the learning results and validated by combining the road connectivity. Experimental results on real Quickbird images demonstrate the effectiveness and robustness of the proposed method.

**2. LITERATURE SURVEY:**

**“Road Extraction in Rural and Urban Areas,”**

An approach for automatic road extraction from digital aerial imagery is presented. The extraction is based on a semantic model for roads. The images are divided into different so-called "global contexts": rural, forest, and urban. Different parts of the road model and different strategies are used in the different global contexts. In rural areas, a multi-scale approach is employed to find initial hypotheses for roadsides, which are then grouped into road segments using local context knowledge. In urban areas, road markings and DEM information are used to extract road segments. In addition to the local grouping, road segments are selected and linked into a global road network. An external evaluation shows the high quality of the results that are obtainable automatically with the proposed approach. 1 INTRODUCTION In the past, the automation of road extraction from digital imagery has received considerable attention. Research on this issue is often motivated by the increasing importanc.

**“Automatic Road Extraction in Urban Scenes and Beyond,”**

In this paper, we present work on automatic road extraction from high resolution aerial imagery taken over urban areas. In order to deal with the high complexity of this type of scenes, we integrate detailed knowledge about roads and their context using explicitly formulated scale-dependent models. The knowledge about how and when certain parts of the road and context model are optimally exploited is condensed in the extraction strategy. To exploit information from multiple views, a fusion strategy for road objects (e.g. lanes) has been developed. It is based on internally computed quality measures and embedded in the system’s concept of self-diagnostic extraction algorithms. The analysis of the final results shows benefits but also remaining deficiencies of this approach. We give an outlook on the utilization of the approach in applications related with traffic monitoring in urban areas.

**“Road Grid Extraction and Verification,”**

While maps exist for most urban areas, there are many locations where the information is not accurate, it may be out of date, or it may be incomplete or of insufficient resolution for the applications. Many difficult problems remain in automated cartography. One of them is the

Extraction of a street grid in an urban environment. Much of the work on road detection has concentrated on low resolution, primarily rural roads (usually producing “spaghetti “roads with no notion of intersections) or high resolution roads without the topological information of the intersections. This paper addresses the problem of extracting a grid with the topological information intact. Given an initial seed intersection, which gives the size and orientation of the regular grid, this system uses a feature-based hypothesis and verify paradigm to ﬁnd the street grid. Veriﬁcation uses local context, provided by an intersection model and by an extended street model, and any available sensors.

**“Cooperative Methods for Road Tracking in Aerial Imagery,”**

A description is given of research in digital mapping and image understanding in the area of automated feature extraction from aerial imagery. The authors discuss a system for road tracking, ARF (A Road Follower) that uses multiple cooperative methods for extracting information about road location and structure from complex aerial imagery. This system is a multilevel architecture for image analysis that follows for cooperation among low-level processes and aggregation of information by high-level analysis components. Two low-level road tracking methods have been implemented: road-surface texture correlation and road-edge following. Each works independently to establish a model of the centerline of the road, its width, and other local properties.

**“Automatic Finding of Main Roads in Aerial Images by Using Geometric-Stochastic Models and Estimation**,”

This paper presents an automated approach to finding main roads in aerial images. The approach is to build geometric-probabilistic models for road image generation. We use Gibbs distributions. Then, given an image, roads are found by MAP (maximum a posteriori probability) estimation. The MAP estimation is handled by partitioning an image into windows, realizing the estimation in each window through the use of dynamic programming, and then, starting with the windows containing high confidence estimates, using dynamic programming again to obtain optimal global estimates of the roads present. The approach is model-based from the outset and is completely different than those appearing in the published literature. It produces two boundaries for each road, or four boundaries when a mid-road barrier is present.

**“An active testing model for tracking roads in satellite images,”**

We present a new approach for tracking roads from satellite images, and thereby illustrate a general computational strategy ("active testing") for tracking 1D structure and other recognition tasks in computer vision. Our approach is related to recent work in active vision on "where to look next" and motivated by the "divide-and-conquer" strategy of parlour games. We choose "tests" (matched filters for short road segments) one at a time in order to remove as much uncertainty as possible about the "true hypothesis" (road position) given the results of the previous tests. The tests are chosen online based on a statistical model for the joint distribution of tests and hypotheses. The problem of minimizing uncertainty (measured by entropy) is formulated in simple and explicit analytical terms. At each iteration new image data are examined and a new entropy minimization problem is solved (exactly), resulting in a new image location to inspect, and so forth. We report experiments using panchromatic SPOT satellite imagery with a ground resolution of ten meters.

**“Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns”**

Presents a theoretically very simple, yet efficient, multiresolution approach to gray-scale and rotation invariant texture classification based on local binary patterns and nonparametric discrimination of sample and prototype distributions. The method is based on recognizing that certain local binary patterns, termed "uniform," are fundamental properties of local image texture and their occurrence histogram is proven to be a very powerful texture feature. We derive a generalized gray-scale and rotation invariant operator presentation that allows for detecting the "uniform" patterns for any quantization of the angular space and for any spatial resolution and presents a method for combining multiple operators for multiresolution analysis. The proposed approach is very robust in terms of gray-scale variations since the operator is, by definition, invariant against any monotonic transformation of the gray scale. Another advantage is computational simplicity as the operator can be realized with a few operations in a small neighborhood and a lookup table. Experimental results demonstrate that good discrimination can be achieved with the occurrence statistics of simple rotation invariant local binary patterns.

**“A Discriminative Feature Space for Detecting and Recognizing Faces,”**

We introduce a novel discriminative feature space which is efficient not only for face detection but also for recognition. The face representation is based on local binary patterns (LBP) and consists of encoding both local and global facial characteristics into a compact feature histogram. The proposed representation is invariant with respect to monotonic gray scale transformations and can be derived in a single scan through the image. Considering the derived feature space, a second-degree polynomial kernel SVM classifier was trained to detect frontal faces in gray scale images. Experimental results using several complex images show that the proposed approach performs favorably compared to the state-of-the-art methods. Additionally, experiments with detecting and recognizing low-resolution faces from video sequences were carried out, demonstrating that the same facial representation can be efficiently used for both detection and recognition.

**“Automatic Main Road Extraction from High Resolution Satellite Imagery,”**

Road information is essential for automatic GIS (geographical information system) data acquisition, transportation and urban planning. Automatic road (network) detection from high resolution satellite imagery will hold great potential for significant reduction of database development/updating cost and turnaround time. From so called low level feature detection to high level context supported grouping, so many algorithms and methodologies have been presented for this purpose. There is not any practical system that can fully automatically extract road network from space imagery for the purpose of automatic mapping. This paper presents the methodology of automatic main road detection from high resolution satellite IKONOS imagery. The strategies include multiresolution or image pyramid method, Gaussian blurring and the line finder using 1-dimemsional template correlation filter, line segment grouping and multi-layer result integration. Multi-layer or multi-resolution method for road extraction is a very effective strategy to save processing time and improve robustness. To realize the strategy, the original IKONOS image is compressed into different corresponding image resolution so that an image pyramid is generated; after that the line finder of 1-dimemsional template correlation filter after Gaussian blurring filtering is applied to detect the road centerline. Extracted centerline segments belong to or do not belong to roads. There are two ways to identify the attributes of the segments, the one is using segment grouping to form longer line segments and assign a possibility to the segment depending on the length and other geometric and photometric attribute of the segment, for example the longer segment means bigger possibility of being road. Perceptual-grouping based method is used for road segment linking by a possibility model that takes multi-information into account; here the clues existing in the gaps are considered. Another way to identify the segments is feature detection back-to-higher resolution layer from the image pyramid.

**“A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting,”**

In the first part of the paper we consider the problem of dynamically apportioning resources among a set of options in a worst-case on-line framework. The model we study can be interpreted as a broad, abstract extension of the well-studied on-line prediction model to a general decision-theoretic setting. We show that the multiplicative weight-update Littlestone–Warmuth rule can be adapted to this model, yielding bonds that are slightly weaker in some cases, but applicable to a considerably more general class of learning problems. We show how the resulting learning algorithm can be applied to a variety of problems, including gambling, multiple-outcome prediction, repeated games, and prediction of points in Rn. In the second part of the paper we apply the multiplicative weight-update technique to derive a new boosting algorithm. This boosting algorithm does not require any prior knowledge about the performance of the weak learning algorithm. We also study generalizations of the new boosting algorithm to the problem of learning functions whose range, rather than being binary, is an arbitrary finite set or a bounded segment of the real line.

**3. SYSTEM ANALYSIS**

**3.1 EXISTING SYSTEM:**

Road extraction in urban areas has been an important task for generating geographic information systems (GIS). Especially in recent years, the rapid development of urban areas makes it urgent to provide up-to-date road maps. The timely road information is very useful for the decision-makers in urban planning, traffic management and car navigation fields, etc.

**Disadvantage:**

1. Less Accuracy

**3.2 PROPOSED SYSTEM:**

In order to deal with the difficulties for building comprehensive road models and to make full use of the characteristics of urban roads, we propose an automatic approach based on machine learning. It can be divided into three steps. First, a series of features reflecting road characteristics are extracted. They include the ratio of bright lines on the road surface, the directional consistency of road markings and local binary patterns (LBP). These features are then input into a learning container, and Ada Boost is adopted to train classifiers and select distinct features. Finally, on the basis of the learning results roads are detected with a sliding window and further validated by combing the road connectivity.

**Advantage:**

1. More Accuracy.

**Modules Information:**

To implement this project we have designed following Modules

1. Upload Satellite Images Dataset: using this module we will upload satellite images dataset to application
2. Run Canny, Hough & LBP Features Extraction Algorithms: using this module we will read all images and then extract features using Canny, Hough and LBP
3. Train AdaBoost Algorithm: using this module we will input extracted features to AdaBoost algorithm to train a model
4. Road Extraction from Test Images: using this module we will input test image and then AdaBoost will learn and extract road from given satellite images

**FUNCTIONAL REQUIREMENTS:**

**SOFTWARE REQIREMENTS:**

**System Attributes:**

1. Frame, parameters

**Data base Requirements:**

No need

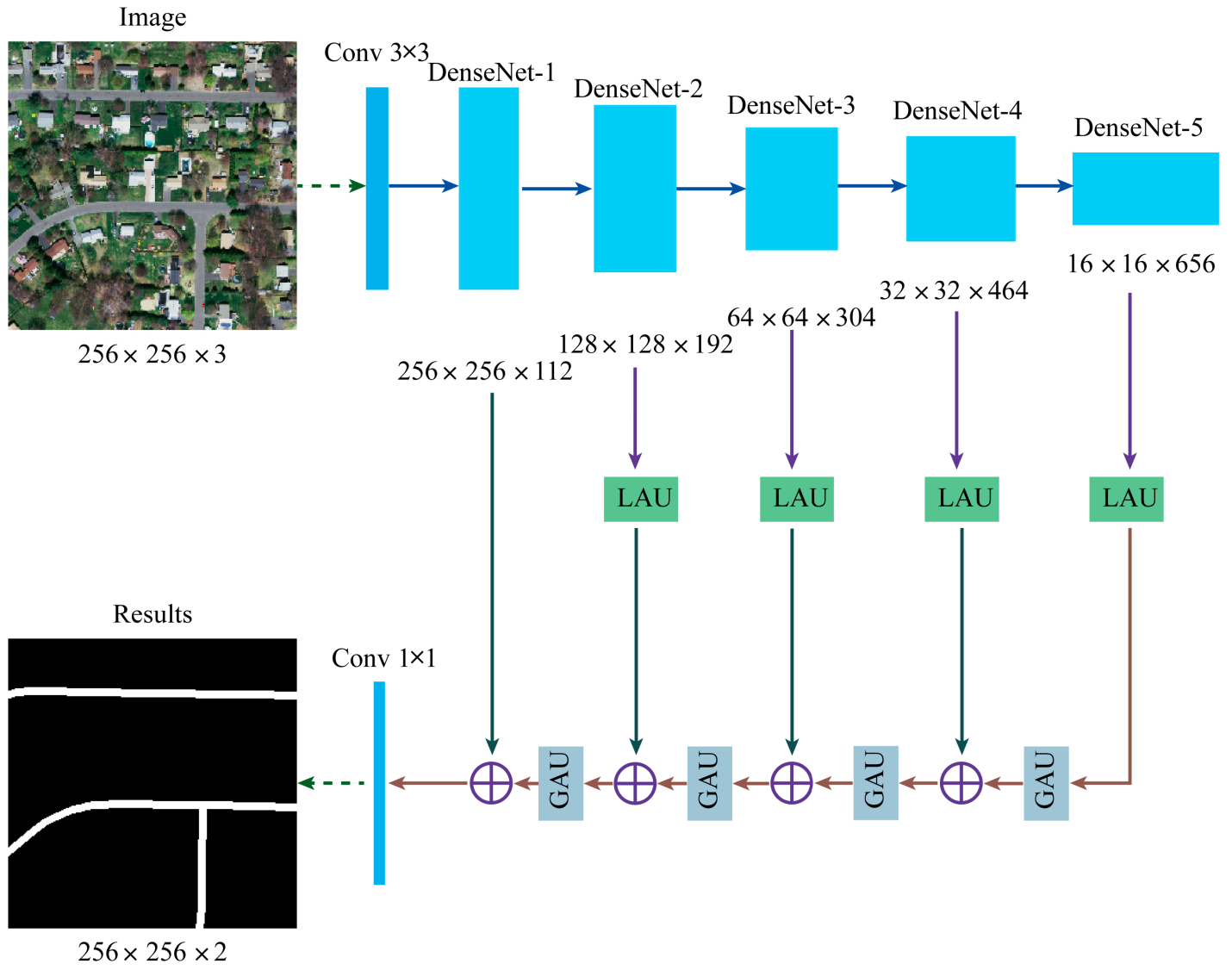
**USECASE:**

Use cases - Use cases describe the interaction between the system and external users that leads to achieving particular goals.

1. Upload Satellite Images Dataset
2. Run Canny, Hough & LBP Features Extraction Algorithms
3. Train AdaBoost Algorithm
4. Road Extraction from Test Images

**User Stories:** In this paper author is using AdaBoost machine learning algorithm to extract road from satellite images. To train AdaBoost author is using Quick Bird satellite images dataset and then applying various features extraction technique such Canny Edge Detection, Hough Line and LBP to extract features from images and then this extracted features will be input to AdaBoost for learning or training a model. This AdaBoost trained model can be applied on any test satellite image to extract road as AdaBoost trained on straight lines features so it can predict straight line road from any test images.

**Work down Structure:**



**Prototype:**

python 3.7.0 or 3.7.4

opencv-python==4.5.1.48

keras==2.3.1

tensor flow==1.14.0

protobuf==3.16.0

h5py==2.10.0

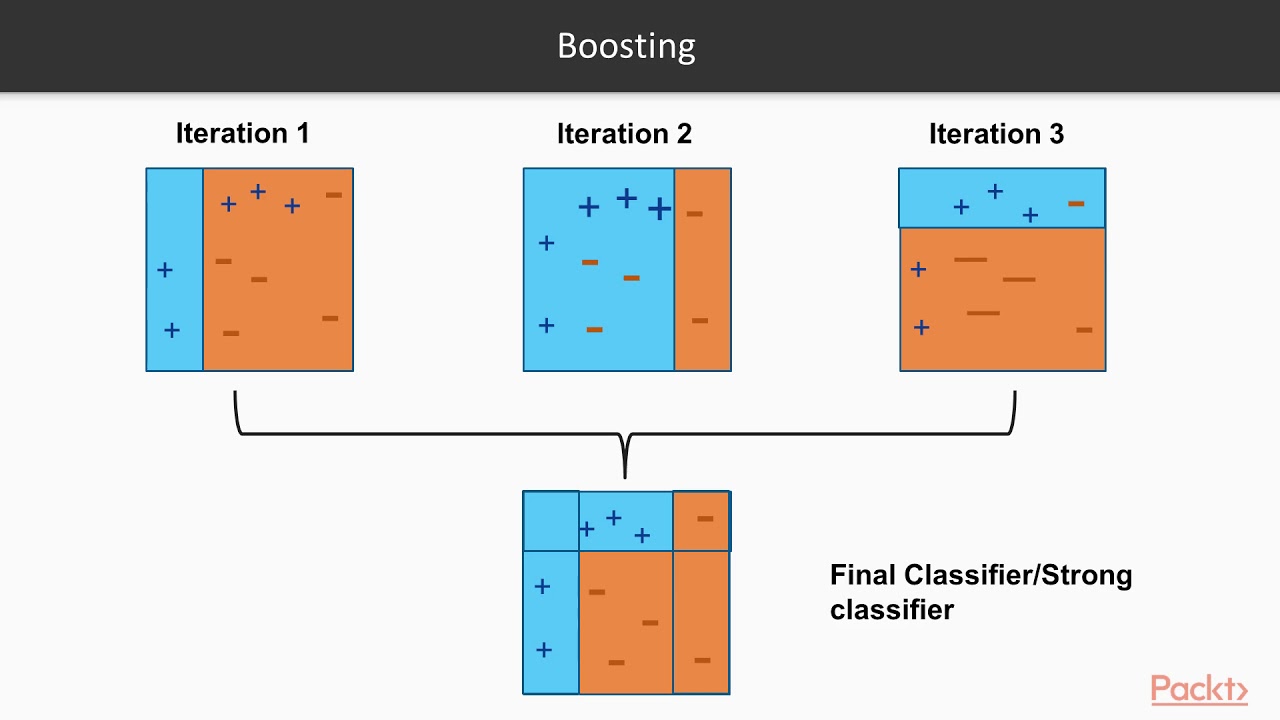
sklearn-extensions==0.0.2

Scikit-learn==0.22.2.post1

Numpy

Pandas

**Models and Diagrams:**



**NON-FUNCTIONAL REQUIREMENT:**

**Usability:**  Usability is a quality attribute that assesses how easy user interfaces are to use. The word "usability" also refers to methods for improving ease-of-use during the design process.(how it was handle entire project easy)

**Security:** the quality or state of being secure: such as. a : freedom from danger : safety. b : freedom from fear or anxiety. c : freedom from the prospect of being laid off job security.

**Readability:** Readability is the ease with which a reader can understand a written text.

**Performance**: the execution of an action. : something accomplished : deed, feat. : the fulfillment of a claim, promise, or request : implementation. 3. : the action of representing a character in a play.

**Availability**: the quality or state of being available trying to improve the availability of affordable housing. 2 : an available person or thing.

**Scalability**: Scalability is the measure of a system's ability to increase or decrease in performance and cost in response to changes in application and system processing demands.

**3.3. PROCESS MODEL USED WITH JUSTIFICATION**

**SDLC (Umbrella Model):**

**Umbrella Activity**

**Umbrella Activity**

**Umbrella Activity**

1. Feasibility Study
2. TEAM FORMATION
3. Project Specification PREPARATION

Business Requirement Documentation

ANALYSIS & DESIGN

CODE

UNIT TEST

DOCUMENT CONTROL

ASSESSMENT

TRAINING

INTEGRATION & SYSTEM TESTING

DELIVERY/INSTALLATION

ACCEPTANCE TEST

Requirements Gathering

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

**Stages in SDLC:**

* Requirement Gathering
* Analysis
* Designing
* Coding
* Testing
* Maintenance

**Requirements Gathering stage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

* Feasibility study is all about identification of problems in a project.
* No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
* Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

**Analysis Stage:**

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

**Designing Stage:**

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

  
When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development (Coding) Stage:**

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

**Integration & Test Stage:**

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.



The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

* **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loa ded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

**3.4. Software Requirement Specification**

**3.4.1. Overall Description**

A Software Requirements Specification (SRS) – a [requirements specification](http://en.wikipedia.org/wiki/Requirements_specification) for a [software system](http://en.wikipedia.org/wiki/Software_system) is a complete description of the behavior of a system to be developed. It includes a set of [use cases](http://en.wikipedia.org/wiki/Use_case) that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. [Nonfunctional requirements](http://en.wikipedia.org/wiki/Non-functional_requirements) are requirements which impose constraints on the design or implementation (such as [performance engineering](http://en.wikipedia.org/wiki/Performance_engineering) requirements, [quality](http://en.wikipedia.org/wiki/Quality_%28business%29) standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A [business analyst](http://en.wikipedia.org/wiki/Business_analyst), sometimes titled [system analyst](http://en.wikipedia.org/wiki/System_analyst), is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the [systems development lifecycle](http://en.wikipedia.org/wiki/Systems_development_life_cycle) domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

* [Business requirements](http://en.wikipedia.org/wiki/Business_requirements) describe in business terms what must be delivered or accomplished to provide value.
* Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
* Process requirements describe activities performed by the developing organization. For instance, process requirements could specify .Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:
* **ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

* **Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

* **TECHNICAL FEASIBILITY**

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to .the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

**3.4.2. External Interface Requirements**

**User Interface**

The user interface of this system is a user friendly python Graphical User Interface.

**Hardware Interfaces**

The interaction between the user and the console is achieved through python capabilities.

**Software Interfaces**

The required software is python.

**SYSTEM REQUIREMENT:**

**HARDWARE REQUIREMENTS:**

# Processor - Intel i3(min)

* Speed - 1.1 GHz
* RAM - 4GB(min)
* Hard Disk - 500 GB
* Key Board - Standard Windows Keyboard
* Mouse - Two or Three Button Mouse
* Monitor - SVGA

**SOFTWARE REQUIREMENTS:**

* Operating System - Windows10(min)
* Programming Language - Python

**4. SYSTEM DESIGN**

**4. SYSTEM DESIGN**

**CLASS DIAGRAM:**

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake

****

**USECASE DIAGRAM:**

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.



**SEQUENCE DIAGRAM:**

A **sequence diagram** is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams**, **event scenarios**, and timing diagrams.



**COLLABORATION DIAGRAM:**

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behaviour of a system.



**COMPONENT DIAGRAM:**

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.



**DEPLOYMENT DIAGRAM:**

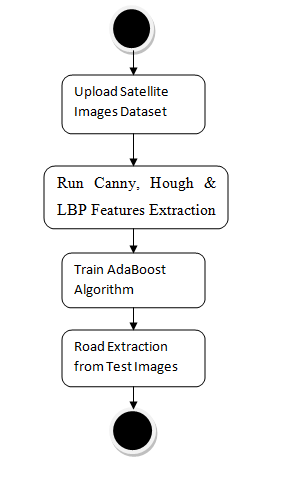
A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.



**ACTIVITY DIAGRAM:**

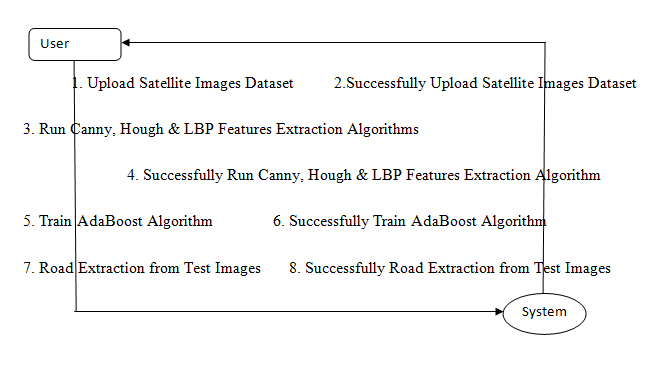
Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent.



**Data flow :**

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.



**5. IMPLEMETATION**

**5.1 Python**

Python is a general-purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D). The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

**History of Python:**

Python is a fairly old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991.

**Why Python was created?**

In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to design of a new language which was later named Python.

**Why the name Python?**

No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late seventies. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

**Features of Python:**

**A simple language which is easier to learn**

Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax.

If you are a newbie, it's a great choice to start your journey with Python.

**Free and open-source**

You can freely use and distribute Python, even for commercial use. Not only can you use and distribute software’s written in it, you can even make changes to the Python's source code.

Python has a large community constantly improving it in each iteration.

**Portability**

You can move Python programs from one platform to another, and run it without any changes.

It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.

**Extensible and Embeddable**

Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code.

This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.

**A high-level, interpreted language**

Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on.

Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.

**Large standard libraries to solve common tasks**

Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server? You can use MySQLdb library using import MySQLdb .

Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.

**Object-oriented**

Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.

With OOP, you are able to divide these complex problems into smaller sets by creating objects.

**Applications of Python:**

**1. Simple Elegant Syntax**

Programming in Python is fun. It's easier to understand and write Python code. Why? The syntax feels natural. Take this source code for an example:

a = 2

b = 3

sum = a + b

print(sum)

**2. Not overly strict**

You don't need to define the type of a variable in Python. Also, it's not necessary to add semicolon at the end of the statement.

Python enforces you to follow good practices (like proper indentation). These small things can make learning much easier for beginners.

**3. Expressiveness of the language**

Python allows you to write programs having greater functionality with fewer lines of code. Here's a link to the source code of Tic-tac-toe game with a graphical interface and a smart computer opponent in less than 500 lines of code. This is just an example. You will be amazed how much you can do with Python once you learn the basics.

**4. Great Community and Support**

Python has a large supporting community. There are numerous active forums online which can be handy if you are stuck.

**5.2 Sample Code:**

**Train.py**

import cv2

import numpy as np

import os

def cannyDetection(image):

edges = cv2.Canny(image,50,150,apertureSize = 3)

return edges

def segmentDetection(img):

height = img.shape[0]

polygons = np.array([[(0, height), (800, height), (380, 290)]])

maskImg = np.zeros\_like(img)

cv2.fillPoly(maskImg, polygons, 255)

segmentImg = cv2.bitwise\_and(img, maskImg)

return segmentImg

def calculateLines(frame, lines):

left = []

right = []

for line in lines:

x1, y1, x2, y2 = line.reshape(4)

parameters = np.polyfit((x1, x2), (y1, y2), 1)

slope = parameters[0]

y\_intercept = parameters[1]

if slope < 0:

left.append((slope, y\_intercept))

else:

right.append((slope, y\_intercept))

left\_avg = np.average(left, axis = 0)

right\_avg = np.average(right, axis = 0)

left\_line = calculateCoordinates(frame, left\_avg)

right\_line = calculateCoordinates(frame, right\_avg)

return np.array([left\_line, right\_line])

def calculateCoordinates(frame, parameters):

slope, intercept = parameters

y1 = frame.shape[0]

y2 = int(y1 - 150)

x1 = int((y1 - intercept) / slope)

x2 = int((y2 - intercept) / slope)

return np.array([x1, y1, x2, y2])

def visualizeLines(frame, lines):

lines\_visualize = np.zeros\_like(frame)

if lines is not None:

for x1, y1, x2, y2 in lines:

cv2.line(lines\_visualize, (x1, y1), (x2, y2), (0, 255, 0), 5)

return lines\_visualize

def get\_pixel(img, center, x, y):

new\_value = 0

try:

if img[x][y] >= center:

new\_value = 1

except:

pass

return new\_value

def lbp\_calculated\_pixel(img, x, y):

center = img[x][y]

val\_ar = []

val\_ar.append(get\_pixel(img, center, x-1, y+1)) # top\_right

val\_ar.append(get\_pixel(img, center, x, y+1)) # right

val\_ar.append(get\_pixel(img, center, x+1, y+1)) # bottom\_right

val\_ar.append(get\_pixel(img, center, x+1, y)) # bottom

val\_ar.append(get\_pixel(img, center, x+1, y-1)) # bottom\_left

val\_ar.append(get\_pixel(img, center, x, y-1)) # left

val\_ar.append(get\_pixel(img, center, x-1, y-1)) # top\_left

val\_ar.append(get\_pixel(img, center, x-1, y)) # top

power\_val = [1, 2, 4, 8, 16, 32, 64, 128]

val = 0

for i in range(len(val\_ar)):

val += val\_ar[i] \* power\_val[i]

return val

X = []

Y = []

label = []

names = []

path = "Dataset/SatelliteImages"

for root, dirs, directory in os.walk(path):

for j in range(len(directory)):

name = os.path.basename(root)

if 'Thumbs.db' not in directory[j]:

image = cv2.imread(root+"/"+directory[j])

canny = cannyDetection(image)

hough = cv2.HoughLinesP(canny, 1, np.pi / 180, 100, np.array([]), minLineLength = 100, maxLineGap = 50)

if hough is not None:

try:

lines = calculateLines(image, hough)

linesVisualize = visualizeLines(image, lines)

output = cv2.addWeighted(image, 0.9, linesVisualize, 1, 1)

height, width, channel = output.shape

img\_gray = cv2.cvtColor(output, cv2.COLOR\_BGR2GRAY)

img\_lbp = np.zeros((height, width,3), np.uint8)

for i in range(0, height):

for m in range(0, width):

img\_lbp[i, m] = lbp\_calculated\_pixel(img\_gray, i, m)

img\_lbp = cv2.resize(img\_lbp, (28, 28))

img\_lbp = img\_lbp.ravel()

img = cv2.imread("Dataset/MaskImages/"+directory[j])

label.append(img)

names.append(directory[j])

lbl = directory[j].split(".")

for k in range(0,10):

X.append(img\_lbp)

Y.append(int(lbl[0]))

print(str(directory[j])+" "+str(lbl))

except Exception:

pass

X = np.asarray(X)

Y = np.asarray(Y)

label = np.asarray(label)

names = np.asarray(names)

np.save("models/X", X)

np.save("models/Y", Y)

np.save("models/label", label)

np.save("models/names", names)

**RoadExtraction.py**

from tkinter import messagebox

from tkinter import \*

from tkinter.filedialog import askopenfilename

from tkinter import simpledialog

import tkinter

import numpy as np

from tkinter import filedialog

import pickle

from sklearn.metrics import accuracy\_score

import cv2

from sklearn.ensemble import AdaBoostClassifier

from sklearn.metrics import accuracy\_score

import os

main = tkinter.Tk()

main.title("Extraction of Main Urban Roads from High Resolution Satellite Images by Machine Learning")

main.geometry("1300x1200")

global filename, X, Y, names, label, adaboost

def cannyDetection(image):

edges = cv2.Canny(image,50,150,apertureSize = 3)

return edges

def segmentDetection(img):

height = img.shape[0]

polygons = np.array([[(0, height), (800, height), (380, 290)]])

maskImg = np.zeros\_like(img)

cv2.fillPoly(maskImg, polygons, 255)

segmentImg = cv2.bitwise\_and(img, maskImg)

return segmentImg

def calculateLines(frame, lines):

left = []

right = []

for line in lines:

x1, y1, x2, y2 = line.reshape(4)

parameters = np.polyfit((x1, x2), (y1, y2), 1)

slope = parameters[0]

y\_intercept = parameters[1]

if slope < 0:

left.append((slope, y\_intercept))

else:

right.append((slope, y\_intercept))

left\_avg = np.average(left, axis = 0)

right\_avg = np.average(right, axis = 0)

left\_line = calculateCoordinates(frame, left\_avg)

right\_line = calculateCoordinates(frame, right\_avg)

return np.array([left\_line, right\_line])

def calculateCoordinates(frame, parameters):

slope, intercept = parameters

y1 = frame.shape[0]

y2 = int(y1 - 150)

x1 = int((y1 - intercept) / slope)

x2 = int((y2 - intercept) / slope)

return np.array([x1, y1, x2, y2])

def visualizeLines(frame, lines):

lines\_visualize = np.zeros\_like(frame)

if lines is not None:

for x1, y1, x2, y2 in lines:

cv2.line(lines\_visualize, (x1, y1), (x2, y2), (0, 255, 0), 5)

return lines\_visualize

def get\_pixel(img, center, x, y):

new\_value = 0

try:

if img[x][y] >= center:

new\_value = 1

except:

pass

return new\_value

def lbp\_calculated\_pixel(img, x, y):

center = img[x][y]

val\_ar = []

val\_ar.append(get\_pixel(img, center, x-1, y+1)) # top\_right

val\_ar.append(get\_pixel(img, center, x, y+1)) # right

val\_ar.append(get\_pixel(img, center, x+1, y+1)) # bottom\_right

val\_ar.append(get\_pixel(img, center, x+1, y)) # bottom

val\_ar.append(get\_pixel(img, center, x+1, y-1)) # bottom\_left

val\_ar.append(get\_pixel(img, center, x, y-1)) # left

val\_ar.append(get\_pixel(img, center, x-1, y-1)) # top\_left

val\_ar.append(get\_pixel(img, center, x-1, y)) # top

power\_val = [1, 2, 4, 8, 16, 32, 64, 128]

val = 0

for i in range(len(val\_ar)):

val += val\_ar[i] \* power\_val[i]

return val

def uploadDataset():

global filename

filename = filedialog.askdirectory(initialdir = "Dataset")

pathlabel.config(text=filename)

text.delete('1.0', END)

text.insert(END,filename+' dataset loaded\n')

def featuresExtraction():

text.delete('1.0', END)

global filename, X, Y, names, label

if os.path.exists("models/X.npy"):

X = np.load("models/X.npy")

Y = np.load("models/Y.npy")

label = np.load("models/label.npy")

names = np.load("models/names.npy")

else:

X = []

Y = []

label = []

names = []

for root, dirs, directory in os.walk(filename):

for j in range(len(directory)):

name = os.path.basename(root)

if 'Thumbs.db' not in directory[j]:

image = cv2.imread(root+"/"+directory[j])

canny = cannyDetection(image)

hough = cv2.HoughLinesP(canny, 1, np.pi / 180, 100, np.array([]), minLineLength = 100, maxLineGap = 50)

if hough is not None:

try:

lines = calculateLines(image, hough)

linesVisualize = visualizeLines(image, lines)

output = cv2.addWeighted(image, 0.9, linesVisualize, 1, 1)

height, width, channel = output.shape

img\_gray = cv2.cvtColor(output, cv2.COLOR\_BGR2GRAY)

img\_lbp = np.zeros((height, width,3), np.uint8)

for i in range(0, height):

for m in range(0, width):

img\_lbp[i, m] = lbp\_calculated\_pixel(img\_gray, i, m)

img\_lbp = cv2.resize(img\_lbp, (28, 28))

img\_lbp = img\_lbp.ravel()

img = cv2.imread("Dataset/SatelliteImages/"+directory[j])

label.append(img)

names.append(directory[j])

lbl = directory[j].split(".")

for k in range(0,10):

X.append(img\_lbp)

Y.append(int(lbl[0]))

print(str(directory[j])+" "+str(lbl))

except Exception:

pass

X = np.asarray(X)

Y = np.asarray(Y)

label = np.asarray(label)

names = np.asarray(names)

for i in range(0,5):

Y[i] = 1000

text.insert(END,"Total satellite images found in dataset : "+str(label.shape[0])+"\n")

text.insert(END,"Total LBP features extracted from each image : "+str(X.shape[1])+"\n\n")

text.insert(END,"LBP Features Extraction process completed")

def trainAdaBoost():

global filename, X, Y, names, label, adaboost

text.delete('1.0', END)

if os.path.exists("models/adaboost.txt"):

with open('models/adaboost.txt', 'rb') as file:

adaboost = pickle.load(file)

file.close()

else:

adaboost = AdaBoostClassifier(n\_estimators=100, random\_state=0)

adaboost.fit(X, Y)

with open('models/adaboost.txt', 'wb') as file:

pickle.dump(adaboost, file)

file.close()

predict = adaboost.predict(X)

completeness = accuracy\_score(Y, predict)

correctness = 1.0 - completeness

text.insert(END,"AdaBoost Learning Process Completed\n\n")

text.insert(END,"Completeness: "+str(completeness)+"\n\n")

text.insert(END,"Correctness: "+str(correctness))

def roadExtraction():

global adaboost

text.delete('1.0', END)

filename = filedialog.askopenfilename(initialdir = "testImages")#uploading image

image = cv2.imread(filename)#reading images from uploaded file

image1 = image

canny = cannyDetection(image)#getting canny image

hough = cv2.HoughLinesP(canny, 1, np.pi / 180, 100, np.array([]), minLineLength = 100, maxLineGap = 50)#applying houghline transform

if hough is not None: #if hough line detected then road straight line is available in image

try:

lines = calculateLines(image, hough) #get road lines

linesVisualize = visualizeLines(image, lines)

output = cv2.addWeighted(image, 0.9, linesVisualize, 1, 1)

height, width, channel = output.shape

img\_gray = cv2.cvtColor(output, cv2.COLOR\_BGR2GRAY)

img\_lbp = np.zeros((height, width,3), np.uint8)

for i in range(0, height):

for m in range(0, width):

img\_lbp[i, m] = lbp\_calculated\_pixel(img\_gray, i, m) #apply LBP on road image part

lbp = img\_lbp

img\_lbp = cv2.resize(img\_lbp, (28, 28))

img\_lbp = img\_lbp.ravel()

temp = []

temp.append(img\_lbp)#add LBP to temp array

temp = np.asarray(temp)#convert array to numpy

predict = adaboost.predict(temp)[0] #predict or learn and then extract road from give images using aDABOOST

lbl = 0

for k in range(len(names)):

if names[k] == str(predict)+".png":

lbl = k

break

print(lbl)

print(predict)

road\_extract = label[lbl]

print("done here")

road\_extract = cv2.cvtColor(road\_extract, cv2.COLOR\_BGR2GRAY)

road\_extract = cv2.bitwise\_and(image1, image1, mask=road\_extract)

print("done 1 here")

cv2.imshow("Satellite Image", image1) #display all road and extracted road images

cv2.imshow("canny Image", canny)

cv2.imshow("LBP Image", lbp)

cv2.imshow("Extracted Road Image", road\_extract)

cv2.waitKey(0)

except Exception:

pass

def close():

main.destroy()

font = ('times', 16, 'bold')

title = Label(main, text='Extraction of Main Urban Roads from High Resolution Satellite Images by Machine Learning')

title.config(bg='dark goldenrod', fg='white')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 13, 'bold')

uploadButton = Button(main, text="Upload Satellite Images Dataset", command=uploadDataset)

uploadButton.place(x=700,y=100)

uploadButton.config(font=font1)

pathlabel = Label(main)

pathlabel.config(bg='DarkOrange1', fg='white')

pathlabel.config(font=font1)

pathlabel.place(x=700,y=150)

featuresButton = Button(main, text="Run Canny, Hough & LBP Features Extraction Algorithms", command=featuresExtraction)

featuresButton.place(x=700,y=200)

featuresButton.config(font=font1)

adaboostButton = Button(main, text="Train AdaBoost Algorithm", command=trainAdaBoost)

adaboostButton.place(x=700,y=250)

adaboostButton.config(font=font1)

extractButton = Button(main, text="Road Extraction from Test Images", command=roadExtraction)

extractButton.place(x=700,y=300)

extractButton.config(font=font1)

exitButton = Button(main, text="Exit", command=close)

exitButton.place(x=700,y=350)

exitButton.config(font=font1)

font1 = ('times', 12, 'bold')

text=Text(main,height=30,width=80)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=10,y=100)

text.config(font=font1)

main.config(bg='turquoise')

main.mainloop()

**6. TESTING:**

**Implementation and Testing:**

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

## Implementation

## The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

## Testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

### System Testing

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to use the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

**Module Testing**

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

**Integration Testing**

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

**Acceptance Testing**

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Case Name** | **Test Case Desc.** | **Test Steps** | | | | **Test Case Status** | **Test Priority** |
| **Step** | **Expected** | | **Actual** |
| 01 | Upload Satellite Images Dataset | Verify  Upload Satellite Images Dataset or not | If Upload Satellite Images Dataset may not Uploaded | we cannot do any further operations | we can do further operations | | High | High |
| 02 | Run Canny, Hough & LBP Features Extraction Algorithms | Verify Run Canny, Hough & LBP Features Extraction Algorithms or not | If Canny, Hough & LBP Features Extraction Algorithms may not be run | we cannot do any further operations | we can do further operations | | High | High |
| 03 | Train AdaBoost Algorithm | Verify  Train AdaBoost Algorithm or not | If AdaBoost Algorithm may not Train | we cannot do any further operations | we can do further operations | | High | High |
| 04 | Road Extraction from Test Images | Verify Road Extraction from Test Images or not | If Road Extraction from Test Images may not be test | we cannot do any further operations | we can do further operations | | High | High |

**7. SCREENSHOTS:**

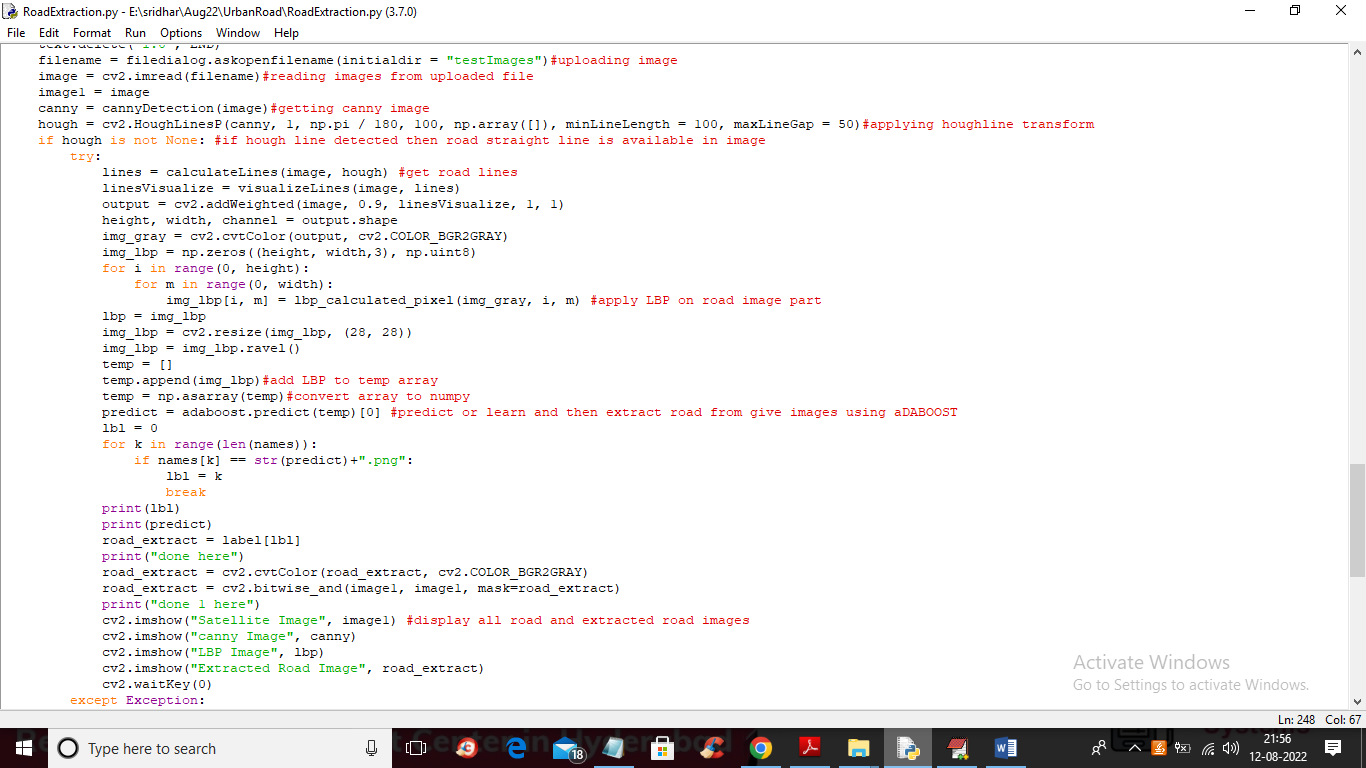
Extraction of Main Urban Roads from High Resolution Satellite Images by Machine Learning

In this paper author is using AdaBoost machine learning algorithm to extract road from satellite images. To train AdaBoost author is using QuickBird satellite images dataset and then applying various features extraction technique such Canny Edge Detection, Hough Line and LBP to extract features from images and then this extracted features will be input to AdaBoost for learning or training a model. This AdaBoost trained model can be applied on any test satellite image to extract road as AdaBoost trained on straight lines features so it can predict straight line road from any test images.

To implement this project we have apply following techniques

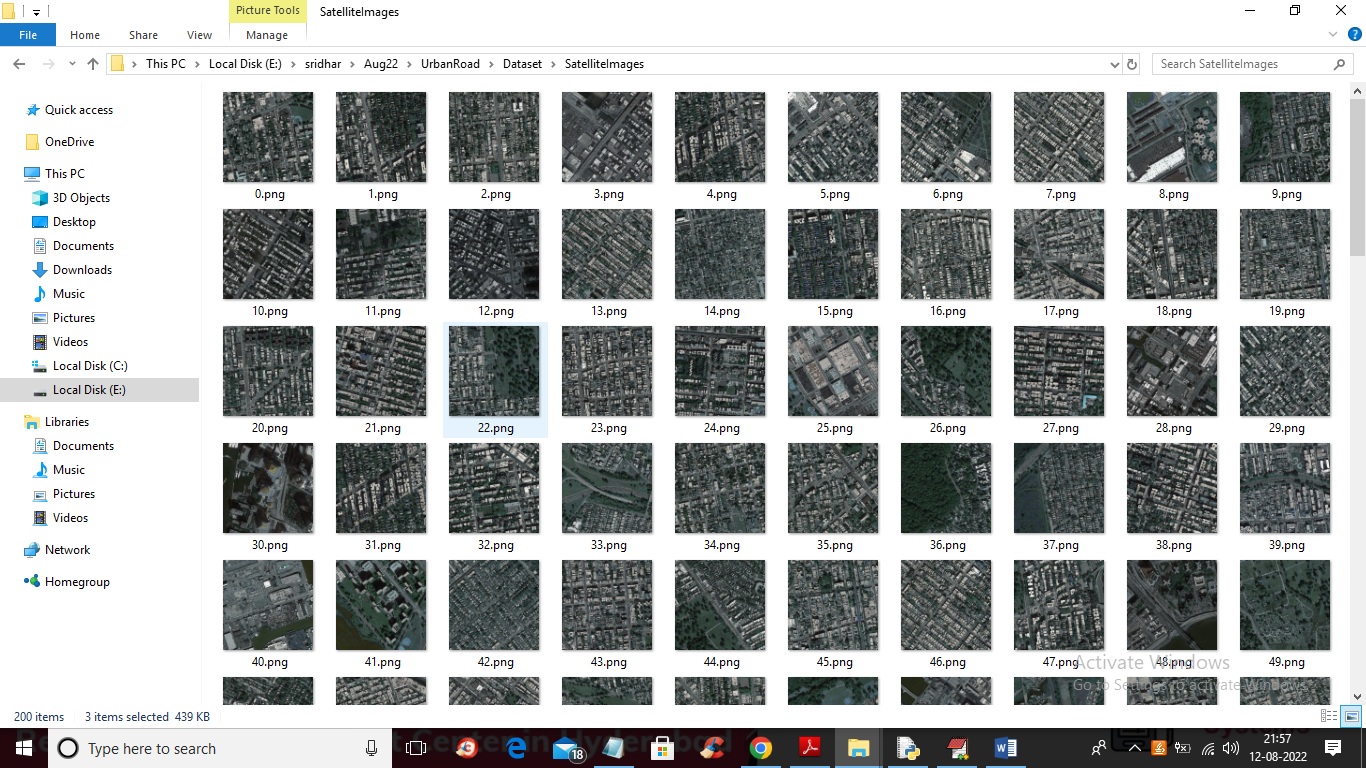
1. Input images: using this module we will input satellite images
2. Canny Edge Detection: using this method we will extract edges from images
3. Hough Transformation: if extracted edges contains straight line then we got road in input images and then extract features
4. LBP: Extracted features will be input to LBP algorithm to extract out road lines from images
5. AdaBoost Learning: extracted LBP features will be input to AdaBoost algorithm to train a model.
6. Road Extraction: AdaBoost trained model will be applied on test image to get road

In below screen we are showing code for above method implementations



In above screen read red colour comments to know about all algorithms used in the paper.

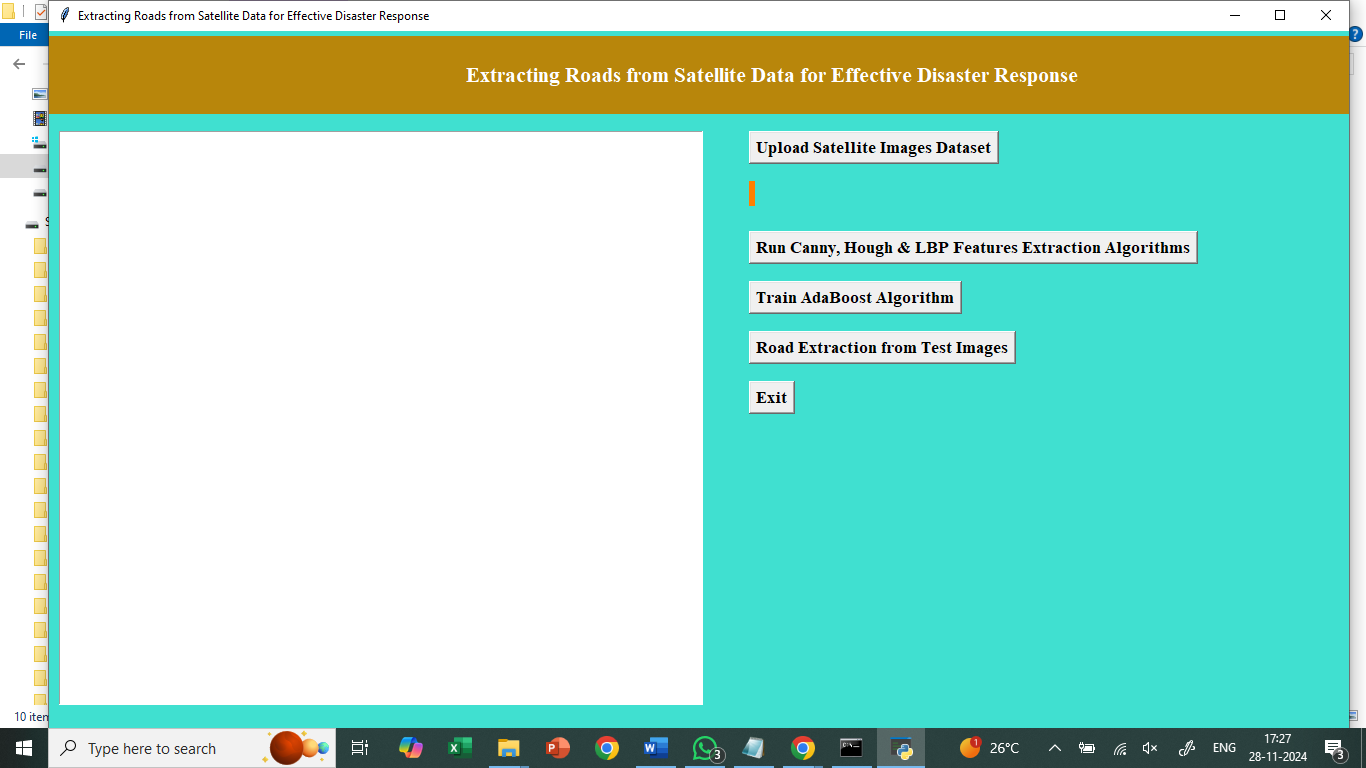
To implement this project we have used below dataset images available inside ‘Dataset’ folder



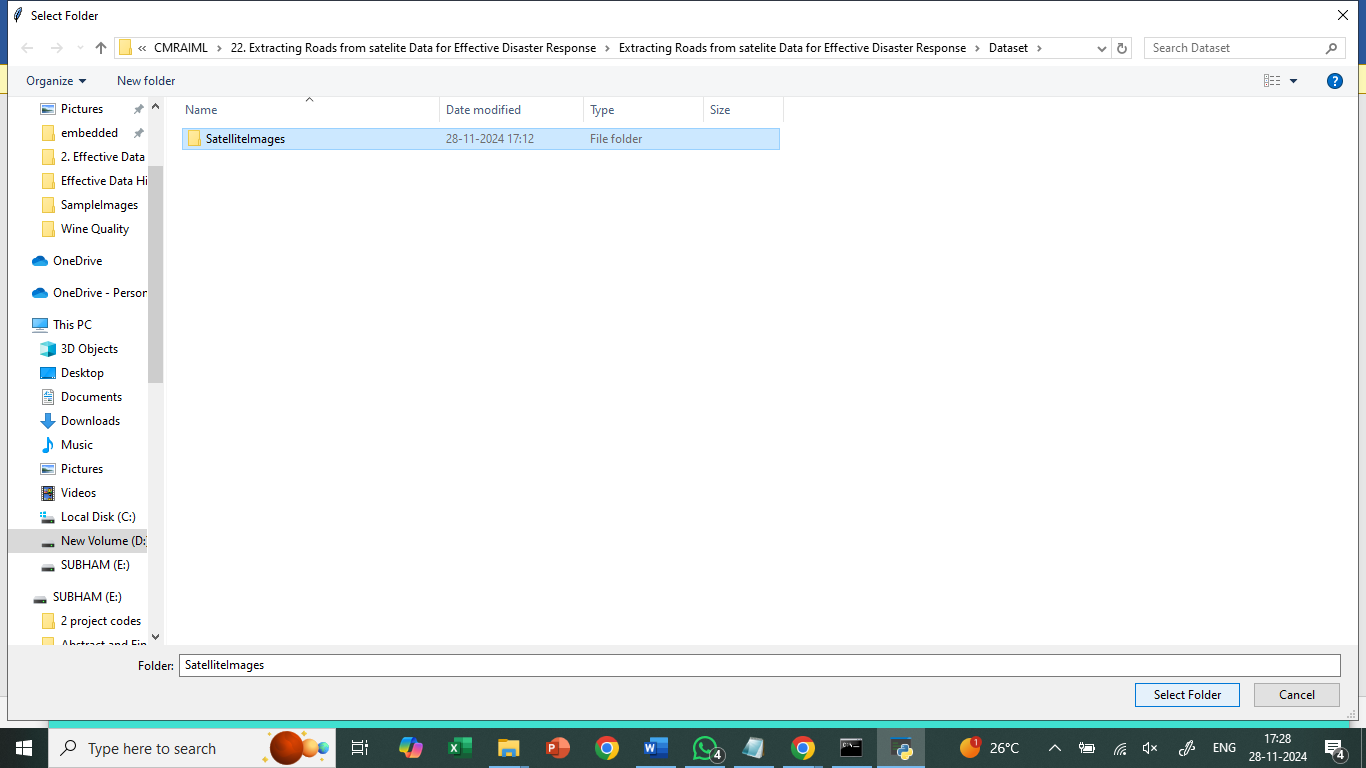
We are using above images to implement this project

SCREEN SHOTS

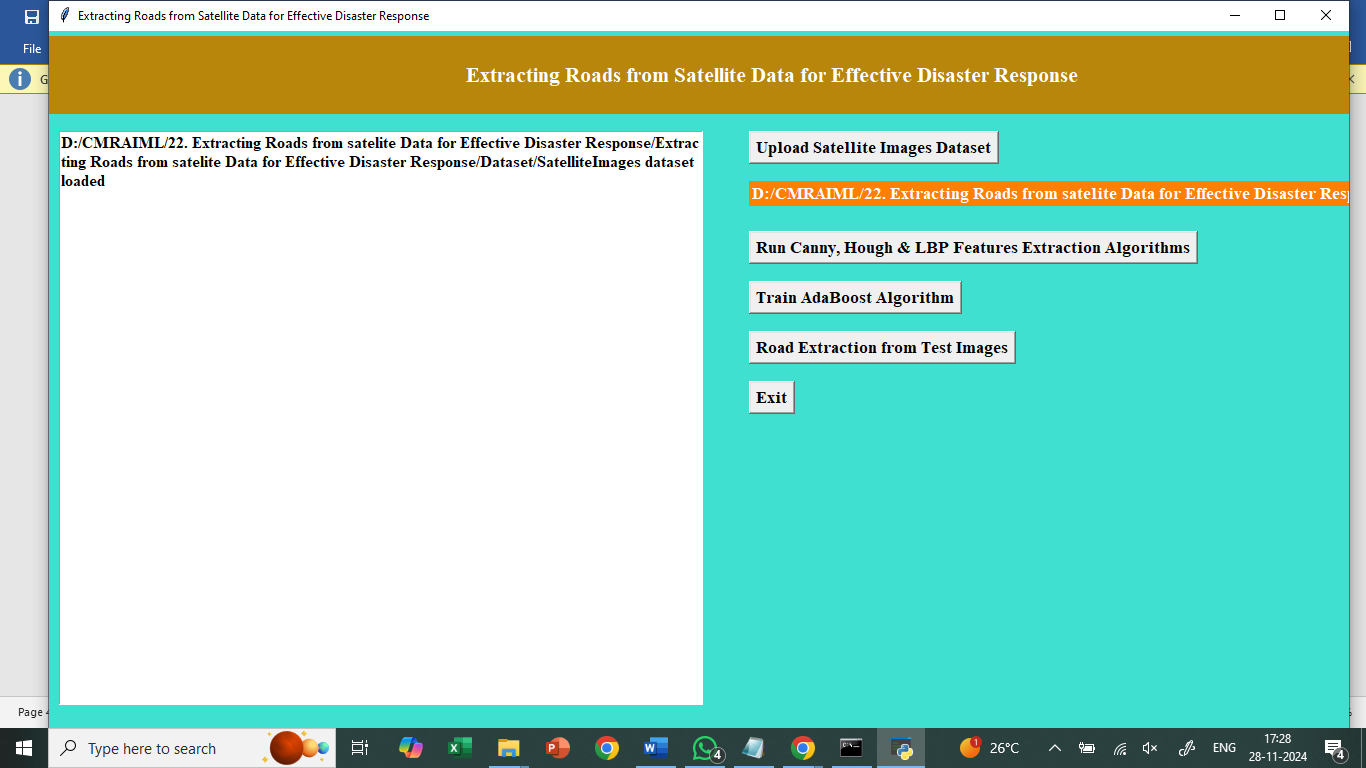
To run project double click on ‘run.bat’ file to get below screen



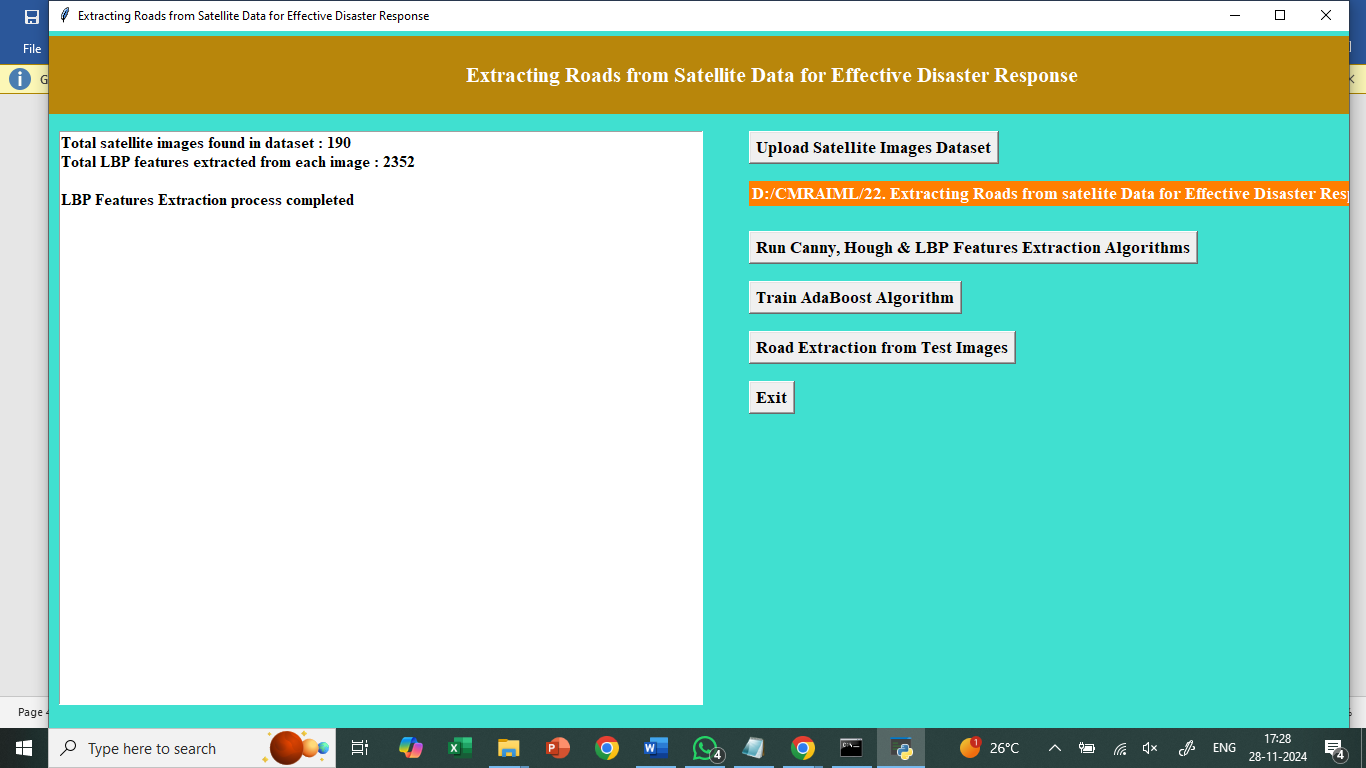
In above screen click on ‘Upload Satellite Images Dataset’ button to upload dataset and get below output



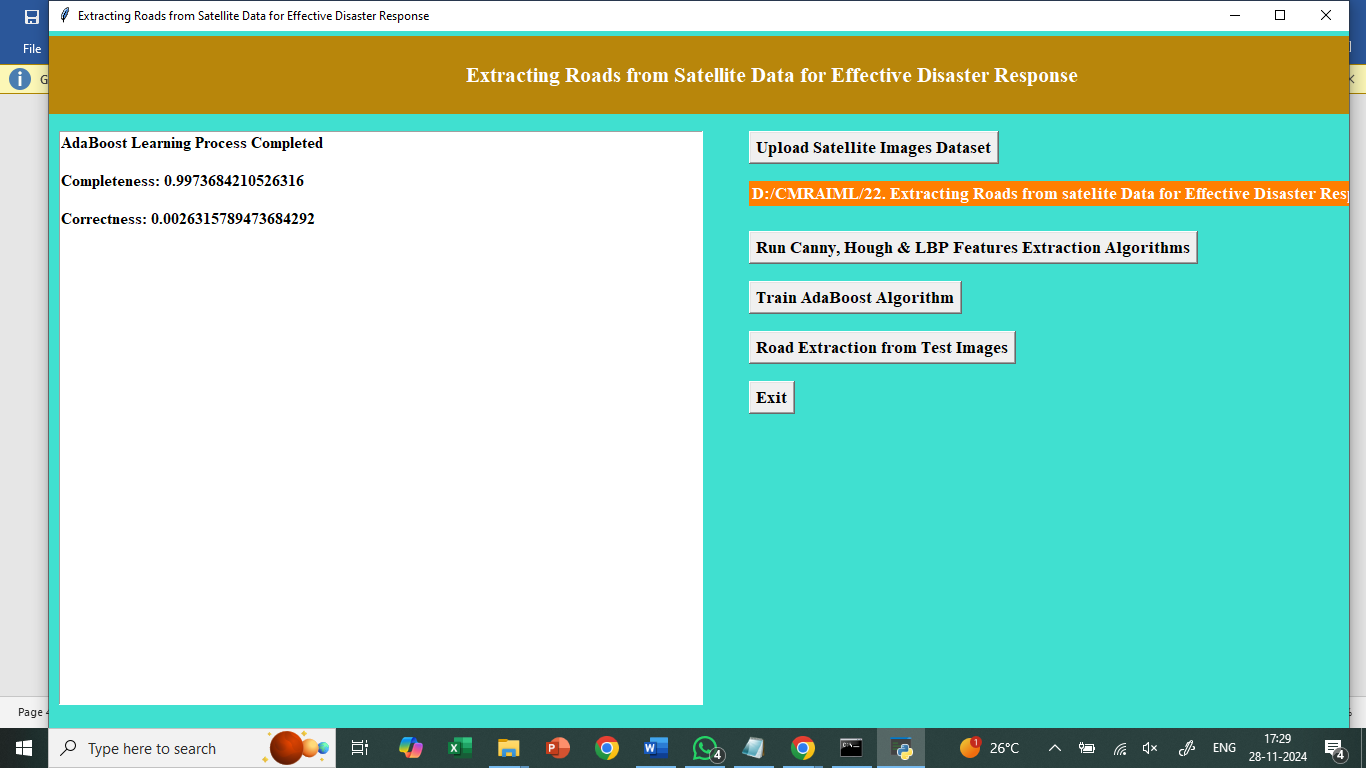
In above screen selecting and uploading ‘Satellite Images’ folder and then click on ‘Select Folder’ button to load dataset and get below output



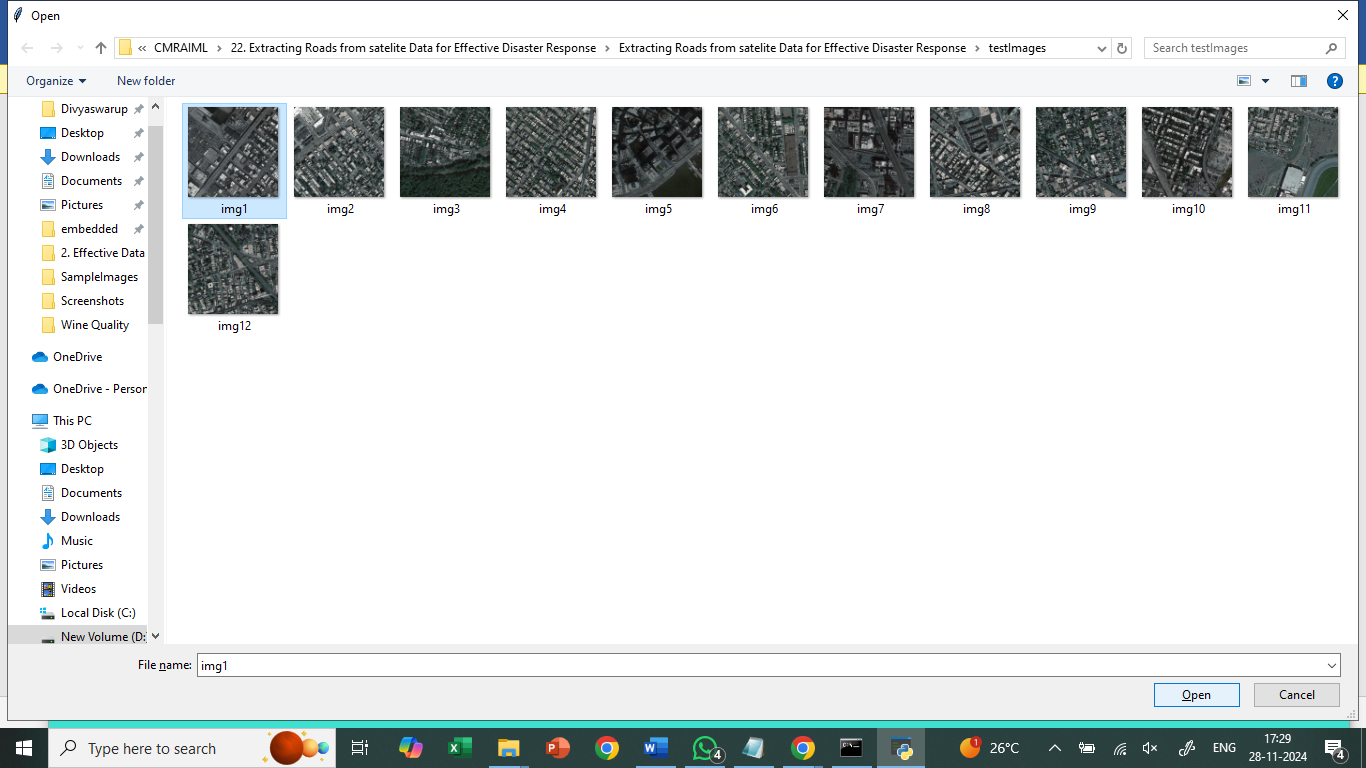
In above screen dataset loaded and now click on ‘Run Canny, Hough & LBP Features Extraction Algorithms’ button to extract features from all dataset images and get below output



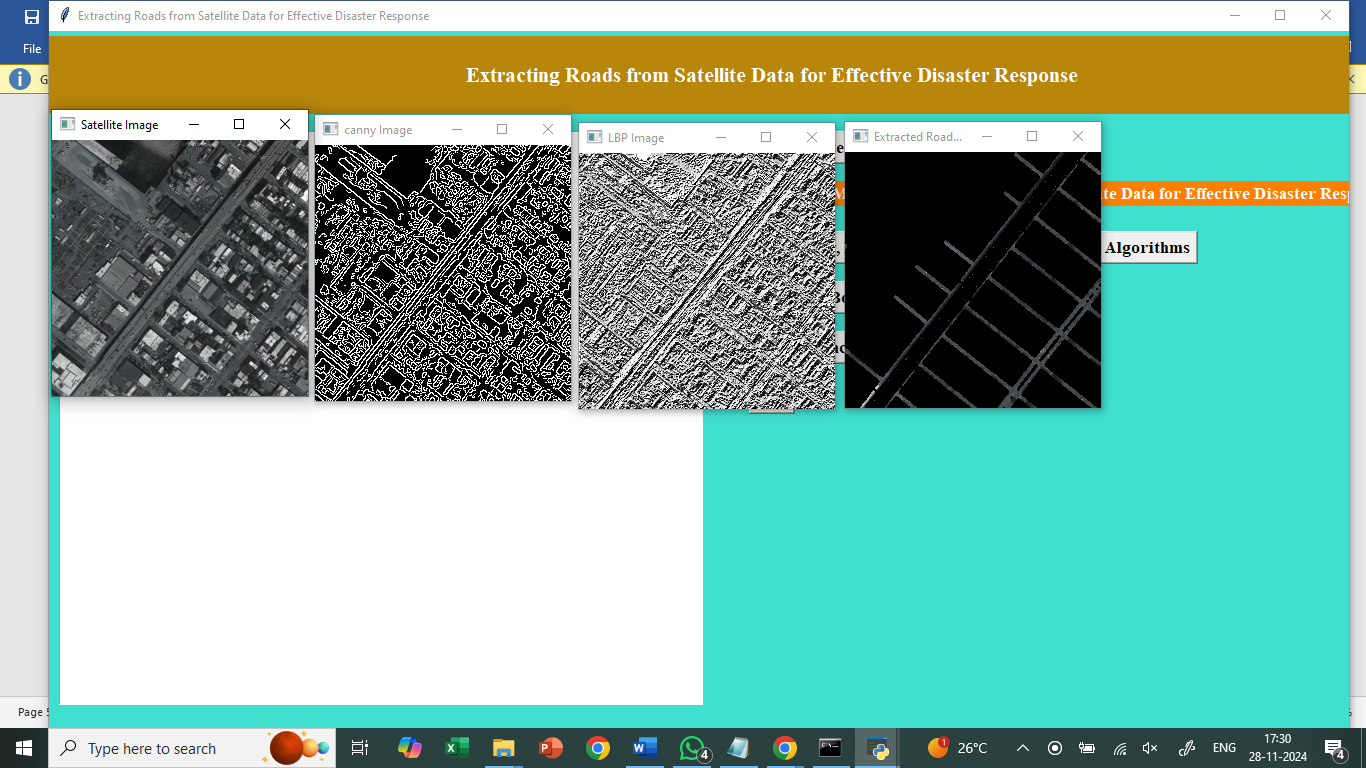
In above screen we can see application found 190 images in dataset and then extract 2352 features from each images and then generate features extracted training array and now click on ‘Train AdaBoost Algorithm’ button to train AdaBoost and get below output



In above screen AdaBoost training completed and we got Completeness (refers to correct prediction %) value as 0.99% and we got Correctness (wrong prediction %) as 0.0026 and now click on ‘Road Extraction from Test Images’ button to upload Satellite image and then AdaBoost will extract road from it



In above screen selecting and uploading ‘img1.png’ file and then click on ‘Open’ button to get below output



In above screen first image is the uploaded Satellite image and second image is Canny Edge detected image and 3rd image is the LBP image and in LBP image we can see straight ROAD line clearly and this line will extract by AdaBoost and give output as 4th image and in 4th image we can see extracted road clearly and in 4th image we can see small white colour dots as vehicles. Similarly you can upload and test other images

**8. CONCLUSION:**

In this paper author is using AdaBoost machine learning algorithm to extract road from satellite images. To train AdaBoost author is using Quick Bird satellite images dataset and then applying various features extraction technique such Canny Edge Detection, Hough Line and LBP to extract features from images and then this extracted features will be input to AdaBoost for learning or training a model. This AdaBoost trained model can be applied on any test satellite image to extract road as AdaBoost trained on straight lines features so it can predict straight line road from any test images

**9. REFERENCES:**

1. S. Hinz, A. Baumgartner, C. Steger, H. Mayer, W. Eckstein, H. Ebner and B. Radig, “Road Extraction in Rural and Urban Areas,” Semantic Modeling for the Acquistion of Topographic Information from Images and Maps, pp. 7-27, 1999.

2. S. Hinz, “Automatic Road Extraction in Urban Scenes and Beyond,” ISPRS, Vol. 35, pp. 349-354, July 2004.

3. K. Price, “Road Grid Extraction and Verification,” International Archives of Photogrammetry and Remote Sensing, Vol. 32, Part 3-2W5, pp. 101-106, 1999.

4. D. M. McKeown and J. L. Denlinger, “Cooperative Methods for Road Tracking in Aerial Imagery,” CVPR, pp. 662-672, 1988.

5. Meir Barzohar and David B. Cooper, “Automatic Finding of Main Roads in Aerial Images by Using Geometric-Stochastic Models and Estimation,” IEEE Trans. PAMI, Vol. 18, No. 7, pp. 707-721, July 1996. 6. Geman, D., B.Jedynak, “An active testing model for tracking roads in satellite images,” IEEE Trans. PAMI, Vol.18, No.1, pp.1-14, January 1996.

7. T. Ojala, M. Pietikainen and T. Maenpaa, “Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns”, IEEE Trans. Pattern Anal. Mach. Intell., Vol. 24, No. 7, pp. 971-987, July 2002.

8. A. Hadid, M. Pietikainen and T. Ahonen, “A Discriminative Feature Space for Detecting and Recognizing Faces,” CVPR, Vol. 2, No. 2, pp. 797-804, 2004.

9. Xiangyun Hu and C.Vincent Tao, “Automatic Main Road Extraction from High Resolution Satellite Imagery,” ISPRS, XXXIV, August 2002.

10. Y. Freund and R. E. Schapire, “A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting,” European Conference on Computational Learning Theory, Springer-Verlag, pp. 23-37, March 1995.

11. P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” CVPR, Vol. 1, pp. 511-518, 2001.

12. C. Steger, “An unbiased detector of curvilinear structures,” IEEE Trans. PAMI, Vol. 20, No. 2, pp. 113-125, 1998.

13. C. Wiedemann, “Automatic Evaluation of Road Networks,” ISPRS Archives, Vol. XXXIV, Part 3/W8, Munich, pp. 17-19, September 2003.