

# **SEGMENTATION OF KIDNEY STONES IN ABDOMINAL X-RAY IMAGES**

**A PROJECT REPORT**

*Submitted by*

**C.GAYATHRI - 510820104006**

**C.SARANYA - 510820104016**

**A.SHANMATHI - 510820104017**

*in partial fulfillment for the award of the degree  
of*

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**GANADIPATHY TULSI'S JAIN ENGINEERING COLLEGE**



**ANNA UNIVERSITY : CHENNAI 600 025**

**MAY 2024**

**ANNA UNIVERSITY : CHENNAI 600 025**

**BONAFIDE CERTIFICATE**

Certified that this project report “**SEGMENTATION OF KIDNEY STONES  
IN ABDOMINAL X-RAY IMAGES**” is the bonafide work of -

**C.GAYATHRI - 510820104006**

**C.SARANYA - 510820104016**

**A.SHANMATHI - 510820104017**

who carried out the project work under my supervision.

**SIGNATURE**

Mrs. S.I. Santhanalakshmi.M.E.,

**HEAD OF THE DEPARTMENT**

Department Of Computer Science  
And Engineering  
GTEC  
Kaniyambadi

**SIGNATURE**

Mrs. S.I. Santhanalakshmi.M.E.,

**SUPERVISOR**

Department Of Computer Science  
And Engineering  
GTEC  
Kaniyambadi

Submitted for the university examination held on \_\_\_\_\_

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

We first offer our deepest gratitude to our **GOD** the almighty who has given us strength and good health during the course of the project.

We are conscious about our indebtedness to our **Managing Trustee Shri.N.Sugalchand Jain**, our Chairman **Shri.P.Pyarelal Jain**, our Secretary **Shri.T.Amarchand Jain**, our Principal **Dr.M.Barathi**, who inspired us to reach greater heights in the pursuits of knowledge.

We articulate our sincere gratitude to our Head of the Department **Mrs.S.I. Santhanalakshmi, M.E.**, who had taught us the way to do a successful project and without whose constant encouragement and whole idea, the project would not have been possible.

We wish to express our sincere thanks to our Internal Guide **Mrs.S.I.Santhanalakshmi, M.E.**, who had helped us a lot and had given valuable suggestions, which made us finish this project in a very successful and neat way.

We wish to express our sincere thanks and honor to our **Project Coordinator Mrs.S.I.Santhanalakshmi.M.E.**, we extend our thanks to **Mr.R.Sudharsan,M.E,Mrs.Y.Mohanapriya,M.E,&Mr.D.Muruganandham M.E.**,for their motivation. We also express our sincere thanks to the entire team of **teaching and non-teaching staff of CSE Department.**

We take this opportunity to thank our parents and friends who have been the source of inspiration and an encouraging factor throughout the project period.

# **Abstract**

Urinary stone detection plays a crucial role in diagnosing and treating urinary tract disorders. This approach utilizes logistic regression for urinary stone detection in medical images. Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image. Nowadays, image processing is among rapidly growing technologies. It forms core research area within engineering and computer science disciplines too. The process involves selecting datasets containing images of urinary stones and healthy urinary tracts, extracting essential features for identifying urinary stone presence, and training a predictive model using logistic regression. The evaluation of the model's performance on test datasets provides insights into its effectiveness in urinary stone detection.

# TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO
	<b>ABSTRACT</b>	II
	<b>LIST OF FIGURES</b>	VI
	<b>LIST OF ABBRIEVATIONS</b>	VII
1.	<b>INTRODUCTION</b>	01
	1.1.Objective	01
	1.2. Overview	01
	1.3.Introduction	01
2.	<b>SYSTEM PROPOSAL</b>	02
	2.1. Existing System	03
	2.1.1 Disadvantages	03
	2.2. Proposed System	04
	2.2.1 Advantages	04
3.	<b>LITERATURE SURVEY</b>	05
4.	<b>SYSTEM DIAGRAMS</b>	10

4.1. Architecture Diagram	10
4.2. Flow Diagram	11
4.3. UML Diagrams	12
4.3.1 Use Case Diagram	12
4.3.2 Er Diagram	13
4.3.3. Sequence Diagram	14
<b>5. IMPLEMENTATION</b>	<b>15</b>
5.1. Modules	15
5.2. Modules Description	15
5.2.1 Image Collection	15
5.2.2 Preparation	15
5.2.3 Feature Extraction	15
5.2.4 Image Splitting	16
5.2.5 LR Model Development	16
5.2.6 Performance Estimation	16

<b>6.</b>	<b>SYSTEM REQUIREMENTS</b>	<b>17</b>
	6.1.Hardware Requirements	17
	6.2.Software Requirements	17
	6.3.Software Description	18
	6.3.1.Java	18
	6.4.Testing Products	23
	6.4.1 Unit Testing	24
	6.4.2 Integration Testing	25
	6.4.3 Testing Techniques	25
	6.4.4 Software Testing Strategies	26
	Validation Testing	
<b>7.</b>	<b>CONCLUSION</b>	<b>27</b>
<b>8.</b>	<b>FUTURE ENHANCEMENT</b>	<b>28</b>
	<b>APPENDIX I</b>	<b>29</b>
	<b>APPENDIX II</b>	<b>44</b>
	<b>REFERENCES</b>	<b>51</b>

## **LIST OF FIGURES**

<b>FIGURE NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
4.1	Architecture Diagram	9
4.2	Entity Relationship Diagram	10
4.2.1	Use Case Diagram	11
4.2.2	Sequence Diagram	11
4.2.3	Class Diagram	12
4.2.4	Activity Diagram	12



## LIST OF ABBREVIATIONS

ABBREVIATIONS	DESCRIPTION
<b>LR</b>	<b>L</b> ogistic <b>R</b> egression
<b>SVM</b>	<b>S</b> imple <b>V</b> ector <b>M</b> achine
<b>NLP</b>	<b>N</b> atural <b>L</b> anguage <b>P</b> rocessing
<b>GBM</b>	<b>G</b> lioblastoma <b>M</b> ultiforme
<b>CAD</b>	<b>C</b> omputer- <b>A</b> ided <b>D</b> etection
<b>CNN</b>	<b>C</b> onvolutional <b>N</b> eural <b>N</b> etworks
<b>SinGAN</b>	<b>S</b> ingle <b>I</b> mage <b>G</b> enerative <b>A</b> dversarial <b>N</b> etwork
<b>GLCM</b>	<b>G</b> ray <b>L</b> evel <b>C</b> o- <b>O</b> ccurrence <b>M</b> atrix
<b>MYSQL</b>	<b>M</b> y <b>S</b> tructured <b>q</b> uery language
<b>JDBC</b>	<b>J</b> ava <b>D</b> ata <b>B</b> ase <b>C</b> onnectivity
<b>J2ME</b>	<b>J</b> ava <b>2</b> <b>M</b> icro <b>E</b> dition
<b>IPFS</b>	<b>I</b> nter <b>P</b> lanetary <b>F</b> ile <b>S</b> ystem
<b>JRE</b>	<b>J</b> ava <b>R</b> untime <b>E</b> nvironment

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 OBJECTIVES**

The main objectives of our urinary stone detection project are as follows:

1. To develop a reliable and accurate method for detecting urinary stones in medical images.
2. To utilize machine learning techniques, specifically logistic regression, for urinary stone detection.
3. To extract relevant features from medical images to characterize the presence of urinary stones.

### **1.2 OVERVIEW**

Logistic regression is a supervised machine learning algorithm that accomplishes binary classification tasks by predicting the probability of an outcome, event, or observation.

### **1.3 INTRODUCTION:**

Urinary stones, also known as kidney stones, are a common urological condition affecting millions of people worldwide. Early detection of urinary stones is crucial for timely intervention and treatment, as they can lead to severe pain and complications if left untreated. Traditional diagnostic methods for urinary stones involve imaging techniques such as X-rays, ultrasounds, and CT scans. Although radiography is not frequently used for stone detection, advantages of this method include relatively lower radiation exposure than CT

imaging and a lower cost than ultrasonography and CT imaging. However, stone detection in plain x-ray images is often difficult for radiologists and other medical doctors because of the following challenges. In radiography, stones and other anatomical structures are projected in a 2D image; hence small stones are difficult to identify due to the overlaps, and some types of stones is poorly visible. Despite advances in imaging technology, manual detection of urinary stones can be time-consuming and subjective, highlighting the need for automated detection methods. The objective of this presentation is to introduce an approach using machine learning for automated urinary stone detection. We propose the pipeline of a cascaded framework based on the U-Net model for the urinary stones segmentation in plain x-ray images. The significant contributions of our work are summarized as follows: 1.) We propose the pipeline of urinary stone segmentation by using two stages of U-Net models, reducing class imbalance and improving segmentation performance. 2.) We utilize the stone-free images by proposing the stone-embedding augmentation implementing during training the second stage U-Net. 3.) We modify the training loss function by implementing the lesion-size reweighting approach, improving the recall rate of small stones.

## **CHAPTER 2**

### **SYSTEM PROPOSAL**

#### **2.1 EXISTING SYSTEM:**

- The traditional methods for urinary stone segmentation in abdominal X-ray images often relied on manual interpretation by radiologists or urologists.
- These methods involved visually inspecting the X-ray images to identify regions of interest corresponding to urinary stones. However, manual segmentation is subjective, time-consuming, and prone to human error.
- Additionally, some older automated segmentation methods utilized thresholding techniques or simple image processing algorithms to distinguish between urinary stones and surrounding tissues based on pixel intensity values.
- While these methods offered some level of automation, they often lacked robustness and accuracy, especially in complex cases with overlapping structures or low contrast. Overall, the limitations of the old methods underscored the need for more advanced and accurate segmentation techniques, leading to the proposal of logistic regression-based segmentation for urinary stones in abdominal X-ray images.

##### **2.1.1 DISADVANTAGES:**

- Sensitivity to image quality
- Limited automation
- Error-prone

## **1.4 PROPOSED SYSTEM:**

The proposed method aims to employ logistic regression for the segmentation of urinary stones in abdominal X-ray images. This involves several key steps:

### **1. Data Selection and Preprocessing:**

Obtain a dataset comprising abdominal X-ray images containing both urinary stones and healthy urinary tracts. Preprocess the images to enhance contrast and remove noise, ensuring optimal input quality for the segmentation model.

### **2. Feature Extraction:**

Extract relevant features from the preprocessed images to characterize urinary stone presence.

### **3. Model Development:**

Develop a logistic regression model trained on the extracted features to distinguish between urinary stones and normal urinary tracts. The model learns the underlying patterns in the data and assigns probabilities to each image region being a urinary stone.

### **4. Segmentation:**

Apply the trained logistic regression model to segment urinary stones within the abdominal X-ray images. This process involves classifying each pixel or region as either belonging to a urinary stone or not.

## **2.2.1 ADVANTAGES:**

- The main advantages of this process is to improve the performance of the classification by minimizing the miss classification rate.
- To stabilize the features by using the appropriate test and training model.
- More feature extraction is implemented.

## **CHAPTER 3**

### **LITERATURE SURVEY**

**[1]TITLE:** TransUNet: Transformers make strong encoders for medical image segmentation.

**AUTHOR:** J. Chen, Y. Lu, Q. Yu, X. Luo, E. Adeli, Y. Wang, L. Lu, A. L. Yuille, and Y. Zhou.

**JOURNAL NAME AND YEAR:** IEEE TRANSACTIONS, 2022.

#### **METHODOLOGY:**

In recent years, the Transformer architecture has demonstrated remarkable success across various natural language processing (NLP) tasks. However, its application to computer vision tasks, particularly medical image segmentation, has gained traction more recently. TransUNet, a hybrid architecture combining Transformers and convolutional neural networks (CNNs), stands out as a promising approach for medical image segmentation tasks. This paper presents an overview of TransUNet's architecture, highlighting its advantages and disadvantages in the context of medical image segmentation.

#### **Advantages:**

- Transformers generate attention maps that provide insights into the regions of the image that are most relevant for segmentation.

#### **Disadvantage:**

- Medical images, often high-resolution and volumetric, can exacerbate this issue, leading to longer training times and higher computational resource

**[2]TITLE:** Urinary Stone Detection on CT Images Using Deep Convolutional Neural Networks: Evaluation of Model Performance and Generalization.

**AUTHOR:** Ansari Parakh, Hyunkwang Lee, Jeong Hyun Lee, Brian H. Eisner, Dushyant V. Sahani<sup>1</sup>, Synho Do.

**JOURNAL NAME AND YEAR:** IEEE TRANSACTIONS, 2022.

**METHODOLOGY:**

This paper proposes a deep convolutional neural network (CNN) approach for detecting urinary stones in unenhanced abdominal CT scans. The model aims to achieve high accuracy and generalizability across different scanners. The proposed approach involves: Cascading two CNNs: The first CNN identifies the urinary tract region, and the second CNN detects the presence of stones within that region. Transfer learning: Pre-trained CNN models (ImageNet and GrayNet) are employed to leverage existing knowledge and improve performance. Evaluation: The model's performance is assessed using various metrics, including accuracy, sensitivity, specificity, and area under the receiver operating characteristic curve (AUC).

**Advantages:**

- The proposed CNN model achieves a high AUC (0.954) for stone detection, indicating its effectiveness in identifying stones accurately.

**Disadvantage:**

- The model's performance is heavily reliant on the quality and quantity of training data.

**[3] TITLE:** Using synthetic training data for deep learning-based GBM segmentation

**AUTHOR:** L. Lindner, D. Narnhofer, M. Weber, C. Gsaxner, M. Kolodziej, and J. Egger,

**JOURNAL NAME AND YEAR:** IEEE TRANSACTIONS, 2023.

**METHODOLOGY:**

Glioblastoma multiform (GBM) segmentation plays a critical role in diagnosis, treatment planning, and prognosis assessment. Deep learning-based segmentation methods have shown promise in automating this task. However, obtaining a large and diverse dataset of annotated GBM images for training these models is challenging due to limited data availability and annotation costs. Synthetic data generation has emerged as a potential solution to address this issue. This paper explores the use of synthetic training data for deep learning-based GBM segmentation and evaluates its effectiveness compared to using solely real data.

**Advantages:**

- Synthetic data augmentation helps prevent over fitting and improves the generalization ability of the segmentation model.

.

**Disadvantage:**

- Inaccurate or unrealistic synthetic data can introduce biases and degrade the performance of the segmentation model.



**[4] TITLE** SinGAN-Seg: Synthetic training data generation for medical image segmentation.

**AUTHOR:** V. Thambawita, P. Salehi, S. A. Sheshkal, S. A. Hicks, H. L. Hammer, S. Parasa, T. D. Lange.

**JOURNAL NAME AND YEAR:** IEEE TRANSACTIONS, 2023.

**METHODOLOGY:**

Medical image segmentation is a critical task in various clinical applications, including disease diagnosis and treatment planning. Deep learning-based segmentation methods have shown remarkable performance but often require large amounts of annotated data for training, which may be scarce and costly to obtain in medical domains. SinGAN-Seg, a variant of the SinGAN (Single Image Generative Adversarial Network) framework, is proposed as a solution for synthetic training data generation tailored specifically for medical image segmentation tasks. This paper presents an overview of SinGAN-Seg and evaluates its effectiveness in generating synthetic training data for medical image segmentation.

**Advantages:**

- Generating synthetic training data eliminates the need for manual annotation of large medical image datasets, significantly reducing annotation costs and accelerating model development.

**Disadvantage:**

- This lack of robustness can limit the applicability of segmentation models in clinical practice.

**[5] TITLE:** Seamless lesion insertion for data augmentation in CAD training

**AUTHOR:** A. Pezeshk, N. Petrick, W. Chen, and B. Sahiner.

**JOURNAL NAME AND YEAR:** IEEE TRANSACTIONS, 2023.

**METHODOLOGY:**

This study introduces a novel approach called "Seamless Lesion Insertion" for enhancing computer-aided detection (CAD) training datasets through data augmentation. By seamlessly integrating synthetic lesions into medical images, this method aims to diversify and enrich the dataset, thereby improving the robustness and generalization of CAD models. Through detailed experimentation and evaluation, we demonstrate the effectiveness of Seamless Lesion Insertion in enhancing CAD training datasets and ultimately improving lesion detection performance in medical imaging applications.

**Advantages:**

- Seamless lesion insertion augments the CAD training dataset with synthetic lesions, thereby increasing the diversity and variability of the data.

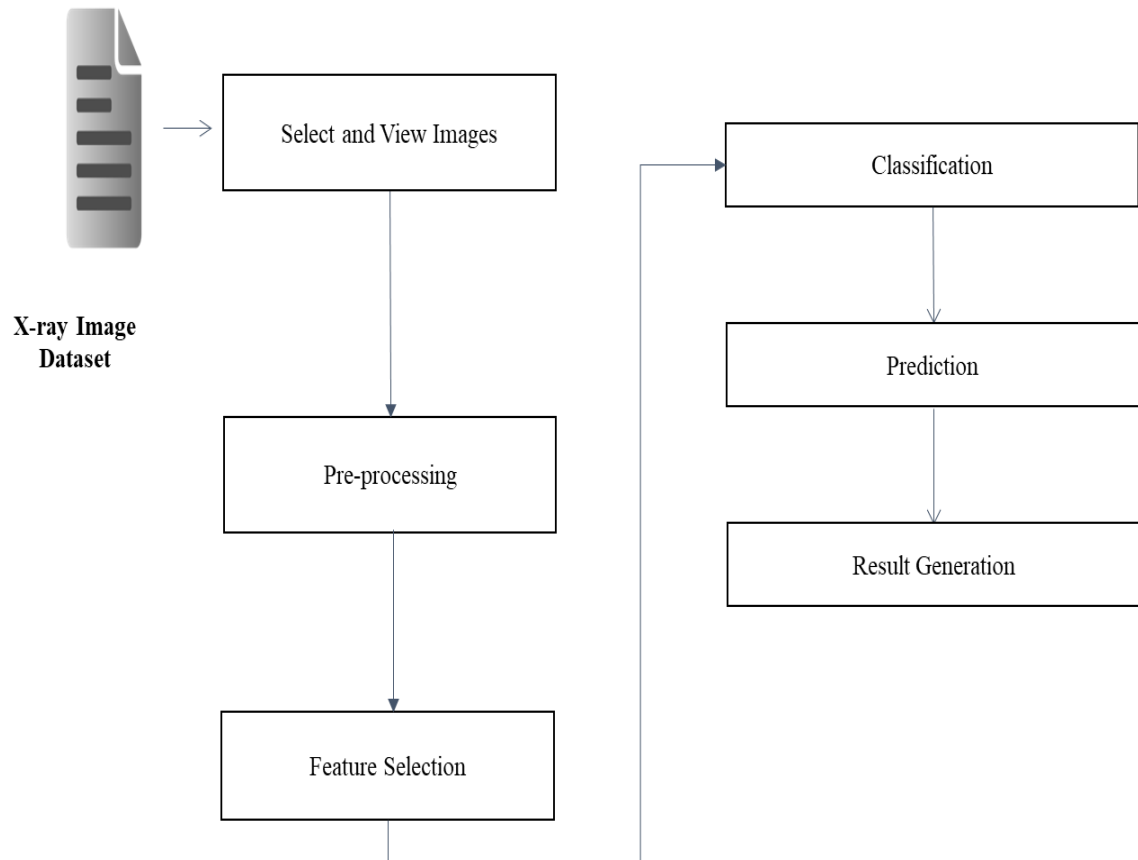
**Disadvantages:**

- While data augmentation can help prevent overfitting by exposing the model to diverse examples, if not properly controlled, it may lead to the generation of unrealistic or irrelevant data that could adversely affect model performance on real-world

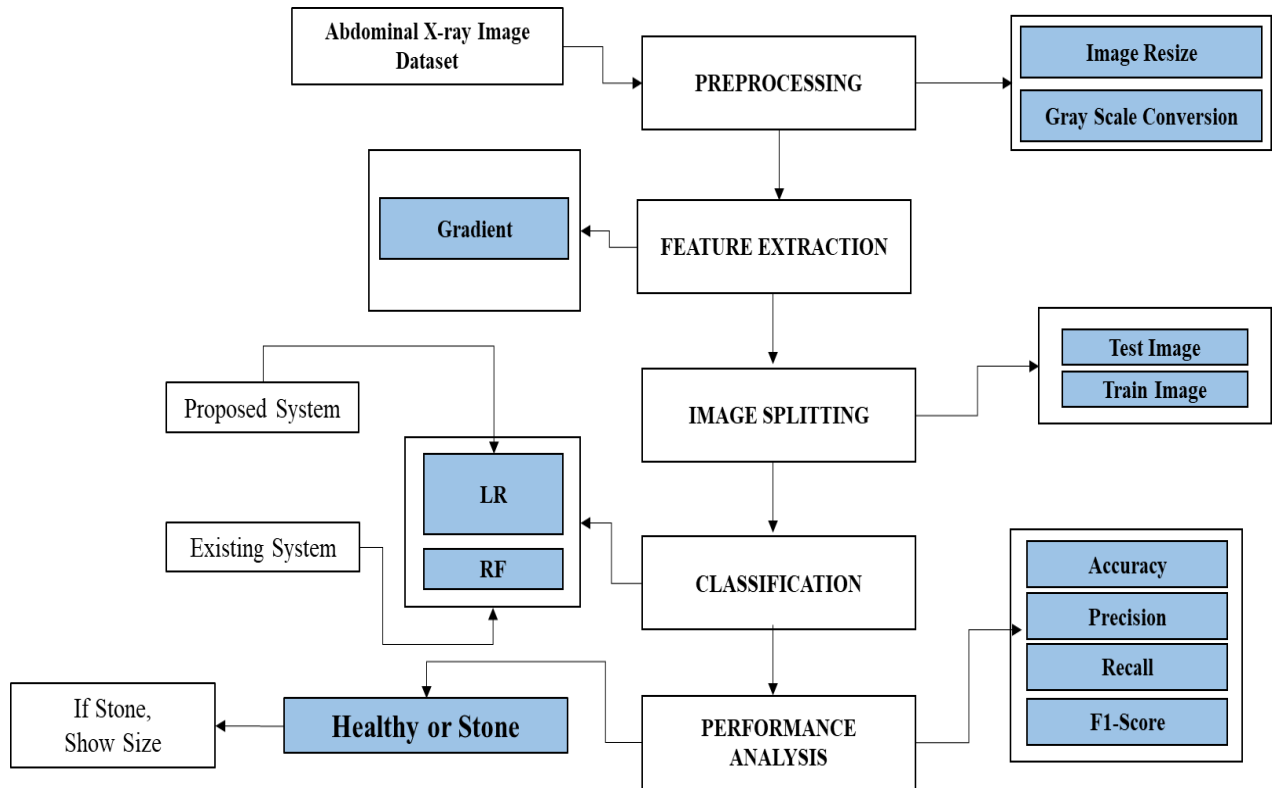
## CHAPTER 4

### SYSTEM DIAGRAMS

#### 4.1 SYSTEM ARCHITECTURE:

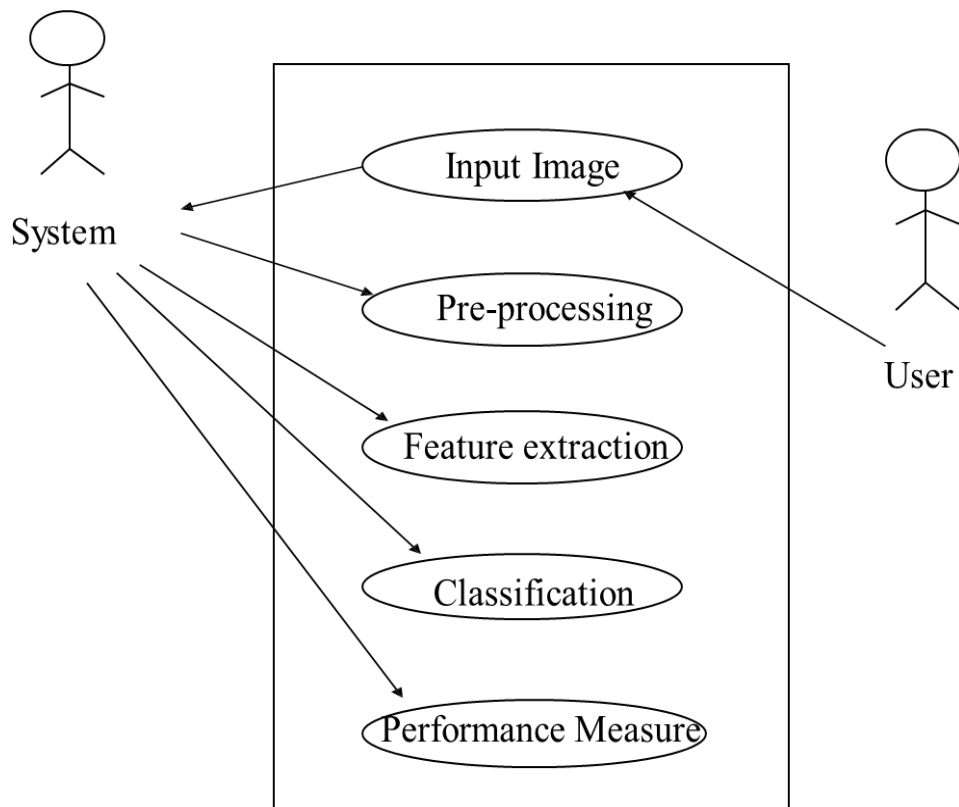


## 4.2 FLOW DIAGRAM

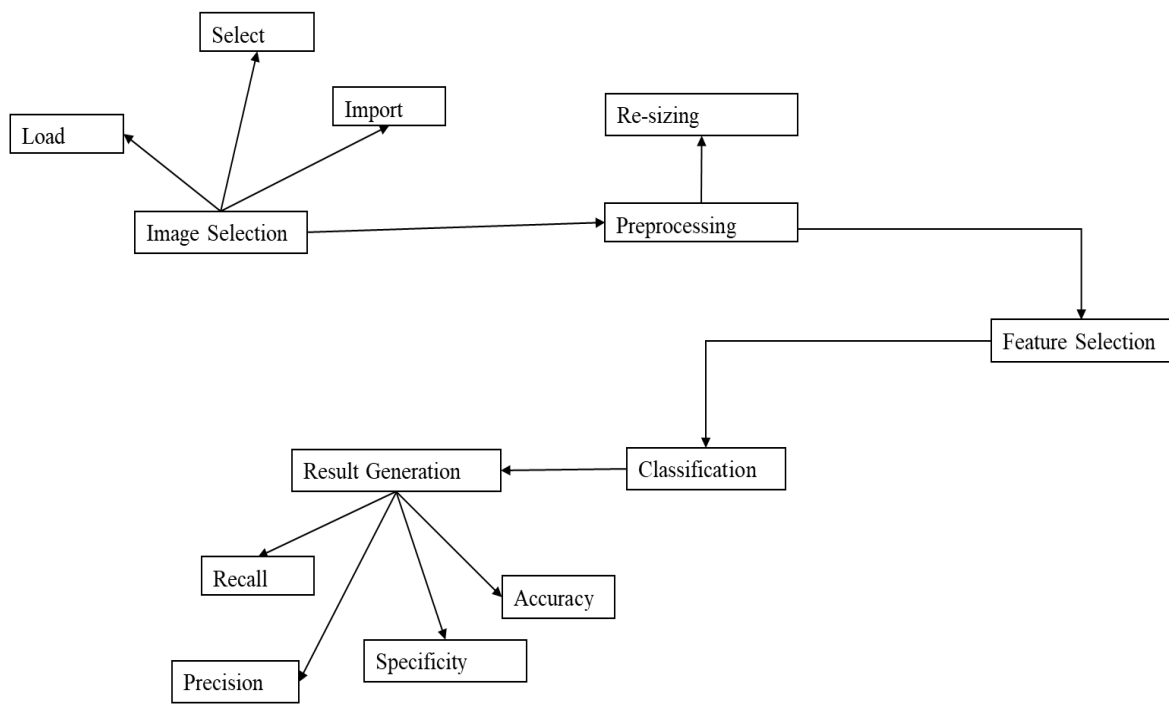


### 4.3 UML DIAGRAMS:

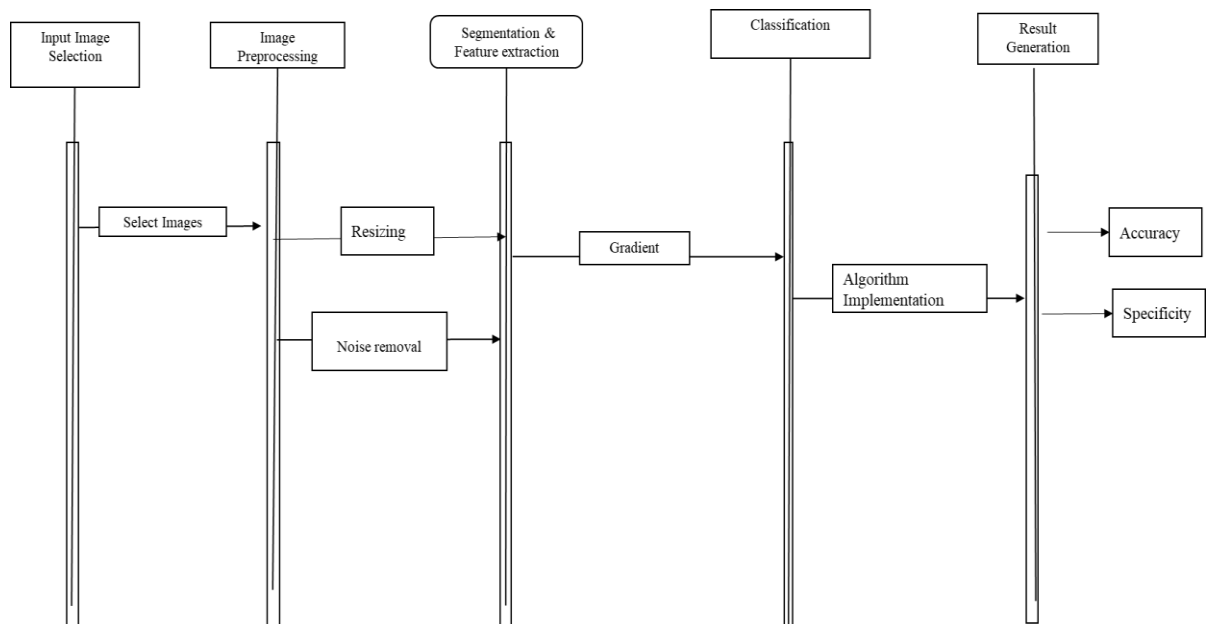
#### 4.3.1 USE CASE DIAGRAM:



### 4.3.2 ER DIAGRAM:



### 4.3.3 SEQUENCE DIAGRAM:



## **CHAPTER 5**

### **IMPLEMENTATION**

#### **5.1 MODULES:**

- Image Collection
- Preparation
- Feature Extraction
- Image Splitting
- LR Model Development
- Performance Estimation

#### **5.2 MODULES DESCRIPTION:**

##### **5.2.1 IMAGE COLLECTION**

- This step is about collecting lots of pictures showing kidney stones from different places like medical databases, patient files, and clinics.
- It's important to have pictures of both healthy kidneys and ones with stones so that the dataset represents different urinary conditions.
- By including all types of images, we can teach our computer models to spot kidney stones accurately, no matter what the kidneys look like.

##### **5.2.2 PREPARATION**

- In this step, we resize all the collected images to a standard size.
- We perform general preprocessing tasks like adjusting brightness and contrast, removing noise, and standardizing color profiles.
- These steps help ensure consistency and improve the quality of the images before further analysis or model training.



### **5.2.3 FEATURE EXTRACTION**

- Feature extraction means finding important details in the images that help tell urinary stones apart from the surrounding tissues.
- This process can be as simple as looking for patterns like shapes, colors, or textures that are common in kidney stones but not in healthy tissue.
- By focusing on these distinctive features, we can teach computers to recognize urinary stones accurately.

### **5.2.4 IMAGE SPLITTING**

- To effectively train and test a machine learning model, it's crucial to split the dataset into two subsets: the training set and the testing set.
- The training set is used to teach the model patterns and relationships between the input data (such as images of urinary stones) and their corresponding labels (e.g., "Healthy" or "Stone").
- The testing set, on the other hand, is used to evaluate how well the model generalizes to new, unseen data.
- By dividing the dataset in this way, we can assess the model's performance accurately and ensure that it can make reliable predictions on real-world data.

### **5.2.5 LR MODEL DEVELOPMENT**

In this step, we create a logistic regression model using the training data.

- This model is like a smart tool that learns from the features we've identified in the images.
- It figures out how to tell if a picture contains a urinary stone or not, based on these features.
- This process helps us build a system that can automatically identify urinary stones accurately.

### 5.2.6 PERFORMANCE ESTIMATION

- After training the logistic regression model, we need to check how well it performs using the testing dataset.
- We use performance metrics like accuracy, precision, recall, and F1 score to evaluate its effectiveness.
- Accuracy tells us how often the model predicts correctly overall, while precision measures how often it's right when it predicts a urinary stone.
- Recall tells us how many urinary stones it finds compared to all the urinary stones in the dataset.
- The F1 score combines precision and recall into a single value, giving us a balanced measure of the model's performance. These metrics help us understand how reliable the model is in identifying urinary stones.

## **CHAPTER 6**

### **SYSTEM REQUIREMENTS**

#### **6.1 HARDWARE REQUIREMENTS:**

- Processor : Intel Core i5
- Hard Disk : 200 GB
- Monitor : 18' LED color
- Mouse : DELL.
- Keyboard : 110 keys enhanced
- RAM : 3GB

#### **6.2 Software Requirements**

- Operating System : Windows 7 / 8 / 10
- Language Used : Java
- User Interface Design : JFrame
- Server : Xampp server

#### **6.3 SOFTWARE DESCRIPTION:**

##### **6.3.1 Java**

Java is a programming language originally developed by James Gosling at Sun Microsystems (now a subsidiary of Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application

developers "write once, run anywhere." Java is currently one of the most popular programming languages in use, particularly for client-server web applications.

- **Java Platform**

One characteristic of Java is portability, which means that computer programs written in the Java language must run similarly on any hardware/operating-system platform. This is achieved by compiling the Java language code to an intermediate representation called Java byte code, instead of directly to platform-specific machine code. Java byte code instructions are analogous to machine code, but are intended to be interpreted by a virtual machine (VM) written specifically for the host hardware.

End-users commonly use a Java Runtime Environment (JRE) installed on their own machine for standalone Java applications, or in a Web browser for Java applets. Standardized libraries provide a generic way to access host-specific features such as graphics, threading, and networking.

A major benefit of using byte code is porting. However, the overhead of interpretation means that interpreted programs almost always run more slowly than programs compiled to native executables would. Just-in-Time compilers were introduced from an early stage that compiles byte codes to machine code during runtime.

Just as application servers such as Glass Fish provide lifecycle services to web applications, the Net Beans runtime container provides them to Swing applications. All new shortcuts should be registered in "Key maps/Net Beans" folder. Shortcuts installed INS Shortcuts folder will be added to all key maps, if there is no conflict. It means that if the same shortcut is mapped to different actions in Shortcut folder and current key map folder (like Key map/Net Beans), the Shortcuts folder mapping will be ignored.

- ✓ Database Explorer Layer API in Database Explorer
- ✓ Loaders-text-dB schema-Actions in Database Explorer
- ✓ Loaders-text-sq.-Actions in Database Explorer
- ✓ Plug-in Registration in Java EE Server Registry

The keyword `public` denotes that a method can be called from code in other classes, or that a class may be used by classes outside the class hierarchy. The class hierarchy is related to the name of the directory in which the `.java` file is located.

The keyword `static` in front of a method indicates a static method, which is associated only with the class and not with any specific instance of that class. Only static methods can be invoked without a reference to an object. Static methods cannot access any class members that are not also static.

The method name `"main"` is not a keyword in the Java language. It is simply the name of the method the Java launcher calls to pass control to the program. Java classes that run in managed environments such as applets and Enterprise JavaBeans do not use or need a `main ()` method. A Java program may contain multiple classes that have main methods, which means that the VM needs to be explicitly told which class to launch from.

The Java launcher launches Java by loading a given class (specified on the command line or as an attribute in a JAR) and starting its `public static void main(String[])` method. Stand-alone programs must declare this method explicitly. The `String []` parameter is an array of `String` objects containing any arguments passed to the class. The parameters to `main` are often passed by means of a command line.

- **Java a High-level Language**

A high-level programming language developed by Sun Microsystems. Java was originally called OAK, and was designed for handheld devices and set-top boxes. Oak was unsuccessful so in 1995 Sun changed the name to Java and modified the language to take advantage of the burgeoning World Wide Web.

Java source code files (files with a .java extension) are compiled into a format called byte code (files with a .class extension), which can then be executed by a Java interpreter. Compiled Java code can run on most computers because Java interpreters and runtime environments, known as Java Virtual Machines (VMs). Byte code can also be converted directly into machine language instructions by a just-in-time compiler (JIT).

Java is a general purpose programming language with a number of features that make the language well suited for use on the World Wide Web. Small Java applications are called Java applets and can be downloaded from a Web server and run on your computer by a Java-compatible Web browser, such as Netscape Navigator or Microsoft Internet Explorer.

Object-Oriented Software Development using Java: Principles, Patterns, and Frameworks contain a much applied focus that develops skills in designing software-particularly in writing well-designed, medium-sized object-oriented programs. It provides a broad and coherent coverage of object-oriented technology, including object-oriented modeling using the Unified Modeling Language (UML) object-oriented design using Design Patterns, and object-oriented programming using Java.

- **Net Beans**

The **Net Beans Platform** is a reusable framework for simplifying the development of Java Swing desktop applications. The Net Beans IDE bundle for

Java SE contains what is needed to start developing Net Beans plug-in and Net Beans Platform based applications; no additional SDK is required.

Applications can install modules dynamically. Any application can include the Update Center module to allow users of the application to download digitally-signed upgrades and new features directly into the running application.

The platform offers reusable services common to desktop applications, allowing developers to focus on the logic specific to their application.

Among the features of the platform are:

- User interface management (e.g. menus and toolbars)
- User settings management
- Storage management (saving and loading any kind of data)
- Window management
- Wizard framework (supports step-by-step dialogs)
- Net Beans Visual Library
- Integrated Development Tools
  
- **J2EE**

A **Java EE application** or a **Java Platform, Enterprise Edition application** is any deployable unit of Java EE functionality. This can be a single Java EE module or a group of modules packaged into an EAR file along with a Java EE application deployment descriptor.

Enterprise applications can consist of the following:

- EJB modules (packaged in JAR files)
- Web modules (packaged in WAR files)

- connector modules or resource adapters (packaged in RAR files)
- Session Initiation Protocol (SIP) modules (packaged in SAR files)
- application client modules
- Additional JAR files containing dependent classes or other components required by the application

## **Wamp Server**

**WAMPs** are packages of independently-created programs installed on computers that use a Microsoft Windows operating system.

Apache is a web server. MySQL is an open-source database. PHP is a scripting language that can manipulate information held in a database and generate web pages dynamically each time content is requested by a browser. Other programs may also be included in a package, such as phpMyAdmin which provides a graphical user interface for the MySQL database manager, or the alternative scripting languages Python or Perl.

## **MySQL**

The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation.

Free-software-open source projects that require a full-featured database management system often use MySQL. Applications which use MySQL databases include: TYPO3, Joomla, WordPress, hob, Drupal and other software built on the LAMP software stack.



## **Platforms and interfaces**

Many programming languages with language-specific APIs include libraries for accessing MySQL databases. These include MySQL Connector/Net for integration with Microsoft's Visual Studio (languages such as C# and VB are most commonly used) and the JDBC driver for Java. In addition, an ODBC interface called Modoc allows additional programming languages that support the ODBC interface to communicate with a MySQL database, such as ASP or ColdFusion. The MySQL server and official libraries are mostly implemented in ANSI C/ANSI C++.

### **6.4 TESTING PRODUCTS:**

System testing is the stage of implementation, which aimed at ensuring that system works accurately and efficiently before the live operation commence. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an error. A successful test is one that answers a yet undiscovered error.

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. . A series of tests are performed before the system is ready for the user acceptance testing. Any engineered product can be tested in one of the following ways. Knowing the specified function that a product has been designed to from, test can be conducted to demonstrate each function is fully operational. Knowing the internal working of a product, tests can be conducted to ensure that “al gears mesh”, that is the internal operation of the product performs according to the specification and all internal components have been adequately exercised.

### **6.4.1 UNIT TESTING:**

Unit testing is the testing of each module and the integration of the overall system is done. Unit testing becomes verification efforts on the smallest unit of software design in the module. This is also known as 'module testing'. The modules of the system are tested separately.

This testing is carried out during the programming itself. In this testing step, each model is found to be working satisfactorily as regard to the expected output from the module. There are some validation checks for the fields. For example, the validation check is done for verifying the data given by the user where both format and validity of the data entered is included. It is very easy to find error and debug the system.

### **6.4.2 INTEGRATION TESTING:**

Data can be lost across an interface, one module can have an adverse effect on the other sub function, when combined, may not produce the desired major function. Integrated testing is systematic testing that can be done with sample data. The need for the integrated test is to find the overall system performance. There are two types of integration testing. They are:

- i) Top-down integration testing.
- ii) Bottom-up integration testing.

### **6.4.3 TESTING TECHNIQUES:**

#### **WHITEBOX TESTING:**

White Box testing is a test case design method that uses the control structure of the procedural design to drive cases. Using the white box testing methods, we Derived test cases that guarantee that all independent paths within a module have been exercised at least once.

#### **BLACK BOX TESTING:**

1. Black box testing is done to find incorrect or missing function
2. Interface error
3. Errors in external database access
4. Performance errors.
5. Initialization and termination errors

In ‘functional testing’, is performed to validate an application conforms to its specifications of correctly performs all its required functions. So this testing is also called ‘black box testing’. It tests the external behaviour of the system. Here the engineered product can be tested knowing the specified function that a product has been designed to perform, tests can be conducted to demonstrate that each function is fully operational.

### **6.4.4 SOFTWARE TESTING STRATEGIES VALIDATION TESTING:**

After the culmination of black box testing, software is completed assembly as a package, interfacing errors have been uncovered and corrected and final series of software validation tests begin validation testing can be defined as many, But a single definition is that validation succeeds when the software functions in a manner that can be reasonably expected by the customer

## **USER ACCEPTANCE TESTING:**

User acceptance of the system is the key factor for the success of the system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system at the time of developing changes whenever required.

## **OUTPUT TESTING:**

After performing the validation testing, the next step is output asking the user about the format required testing of the proposed system, since no system could be useful if it does not produce the required output in the specific format. The output displayed or generated by the system under consideration. Here the output format is considered in two ways. One is screen and the other is printed format. The output format on the screen is found to be correct as the format was designed in the system phase according to the user needs. For the hard copy also output comes out as the specified requirements by the user. Hence the output testing does not result in any connection in the system.

## **CHAPTER 7**

### **CONCLUSION**

- In conclusion, this will investigated the application of logistic regression for urinary stones segmentation in abdominal X-ray images.
- Through the development and evaluation of a logistic regression-based model, we have demonstrated promising results in accurately identifying urinary stones within the images.
- The proposed methodology, which involved data preparation, feature extraction, model development, and evaluation, has shown potential in enhancing urinary stone detection using a statistical learning approach.

## **CHAPTER 8**

### **FUTURE ENHANCEMENT**

- Explore the integration of deep learning architectures such as convolutional neural networks (CNNs) for more complex feature extraction and segmentation tasks.
- Develop real-time urinary stone segmentation systems for clinical use, considering speed, efficiency, and usability in medical settings.
- Extend the segmentation framework to automatically quantify urinary stone characteristics such as size, shape, and location, facilitating more comprehensive diagnostic and treatment planning processes.

## APPENDIX I

### SAMPLE CODING

```
package urinary_stone_detection;

import java.awt.Graphics2D;

import java.awt.Image;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import javax.imageio.ImageIO;

import javax.swing.ImageIcon;

import javax.swing.JFileChooser;

import javax.swing.JOptionPane;

import javax.swing.filechooser.FileNameExtensionFilter;

public class Preprocessing extends javax.swing.JFrame {

    public Preprocessing() {

        initComponents();

    }

    public String path = "./Dataset/";

    private File input;

    public static String imgname, imgpath;

    String Stone_path = path + "Stone";

    String Preprocessed_HealthyPath = "./Preprocessed/Healthy";
```

```
String Healthy_path = path + "Healthy";
```

by the Form Editor.

```
@SuppressWarnings("unchecked")
```

```
// <editor-fold defaultstate="collapsed" desc="Generated Code">
```

```
private void initComponents()
```

```
jPanel1 = new javax.swing.JPanel();
```

```
jLabel3 = new javax.swing.JLabel();
```

```
jLabel1 = new javax.swing.JLabel();
```

```
jButton2 = new javax.swing.JButton();
```

```
jButton3 = new javax.swing.JButton();
```

```
jScrollPane1 = new javax.swing.JScrollPane();
```

```
first = new javax.swing.JLabel();
```

```
jScrollPane2 = new javax.swing.JScrollPane();
```

```
second = new javax.swing.JLabel();
```

```
jScrollPane3 = new javax.swing.JScrollPane();
```

```
third = new javax.swing.JLabel();
```

```
jButton4 = new javax.swing.JButton();
```

```
jButton5 = new javax.swing.JButton();
```

```
jLabel2 = new javax.swing.JLabel();
```

```
jLabel4 = new javax.swing.JLabel();
```



```
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
```

```
jPanel1.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
```

```
jLabel3.setFont(new java.awt.Font("Times New Roman", 1, 24)); //
```

NOI18N

```
jLabel3.setForeground(new java.awt.Color(255, 255, 255));
```

```
jLabel3.setText("URINARY STONES SEGMENTATION IN ");
```

```
jPanel1.add(jLabel3,new
```

```
org.netbeans.lib.awtextra.AbsoluteConstraints(26, 20, -1, -1));
```

```
jLabel1.setFont(new java.awt.Font("Times New Roman", 1, 24)); //
```

NOI18N

```
jLabel1.setForeground(new java.awt.Color(255, 255, 255));
```

```
jLabel1.setText("IMAGE PREPROCESSING");
```

```
jPanel1.add(jLabel1,new
```

```
org.netbeans.lib.awtextra.AbsoluteConstraints(590, 70, -1, -1));
```

```
jButton2.setFont(new java.awt.Font("Times New Roman", 0, 14)); //
```

NOI18N

```
jButton2.setText("Resized Healthy Images");
```

```
jButton2.addActionListener(new java.awt.event.ActionListener() {
```

```

        public void actionPerformed(java.awt.event.ActionEvent evt) {

            jButton2ActionPerformed(evt);

        }

    });

    jPanel1.add(jButton2,

new org.netbeans.lib.awtextra.AbsoluteConstraints(210, 130, 200, 38));

jButton3.setFont(new java.awt.Font("Times New Roman", 0, 14)); //
NOI18N
jButton3.setText("Resized Stone Images");

jButton3.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        jButton3ActionPerformed(evt);

    }

});

jPanel1.add(jButton3,

new org.netbeans.lib.awtextra.AbsoluteConstraints(470, 130, 200, 38));

jScrollPane1.setViewportView(first);

jPanel1.add(jScrollPane1,

new org.netbeans.lib.awtextra.AbsoluteConstraints(20, 220, 250, 220));

```

```

jScrollPane2.setViewportViewView(second);

jPanel1.add(jScrollPane2,

new org.netbeans.lib.awtextra.AbsoluteConstraints(320, 220, 260, 220));

jScrollPane3.setViewportViewView(third);

jPanel1.add(jScrollPane3,

new org.netbeans.lib.awtextra.AbsoluteConstraints(630, 220, 250, 220));

jButton4.setFont(new java.awt.Font("Times New Roman", 0, 14)); //
NOI18N

jButton4.setText("Resize");

jButton4.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        jButton4ActionPerformed(evt);

    }

});

jPanel1.add(jButton4,

new org.netbeans.lib.awtextra.AbsoluteConstraints(30, 130, 130, 40));

jButton5.setFont(new java.awt.Font("Times New Roman", 0, 14)); //
NOI18N

jButton5.setText("Next");

jButton5.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        jButton5ActionPerformed(evt);

    }

});

```

```

        }

    });

    jPanel1.add(jButton5,

new org.netbeans.lib.awtextra.AbsoluteConstraints(750, 130, 130, 40));

    jLabel2.setFont(new java.awt.Font("Times New Roman", 1, 24)); //
NOI18N

    jLabel2.setForeground(new java.awt.Color(255, 255, 255));

    jLabel2.setText(" ABDOMINAL X-RAY IMAGES");

    jPanel1.add(jLabel2,

new org.netbeans.lib.awtextra.AbsoluteConstraints(26, 70, -1, -1));

    jLabel4.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/Image/home.jpg"))); //
NOI18N

    jLabel4.setText("jLabel4");

    jPanel1.add(jLabel4,

new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 920, 490));

    javax.swing.GroupLayout layout =

new javax.swing.GroupLayout(getContentPane());

    getContentPane().setLayout(layout);

    layout.setHorizontalGroup(

```

```
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
            .addComponent(jPanel1,  
        javax.swing.GroupLayout.DEFAULT_SIZE,  
        javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)  
    );
```

```
        layout.setVerticalGroup(  
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
            .addComponent(jPanel1,  
        javax.swing.GroupLayout.DEFAULT_SIZE,  
        javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)  
    );
```

```
    pack();
```

```
    setLocationRelativeTo(null);
```

```
    }// </editor-fold>
```

```
    private static BufferedImage resizeImage(BufferedImage originalImage,  
    int newWidth, int newHeight) {
```

```
        BufferedImage resizedImage = new BufferedImage(newWidth,  
        newHeight, BufferedImage.TYPE_INT_RGB);
```

```
        Graphics2D g = resizedImage.createGraphics();
```

```
        g.drawImage(originalImage.getScaledInstance(newWidth,  
        newHeight, Image.SCALE_SMOOTH), 0, 0, newWidth, newHeight, null);
```

```
        g.dispose();
```

```
        return resizedImage;
```

```

    }

    private static void resizeAndMoveImage(File sourceImage, String
destinationDirectory) throws IOException {

        BufferedImage originalImage = ImageIO.read(sourceImage);

        int newWidth = 200;

        int newHeight = 200;

        BufferedImage    resizedImage    =    resizeImage(originalImage,
newWidth, newHeight);

        String fileName = sourceImage.getName();

        int pos = fileName.lastIndexOf(".");

        String fileNameWithoutExtension = pos > 0 ? fileName.substring(0,
pos) : fileName;

        File newFile = new File(destinationDirectory + File.separator +
fileNameWithoutExtension + ".png");

        ImageIO.write(resizedImage, "png", newFile);

    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)
{

        try {

            ImageIcon imageIcon7 =

            new ImageIcon("./Preprocessed/Healthy/H(1).png");

            first.setIcon(imageIcon7);

```

```

        ImageIcon imageIcon8 =
new ImageIcon("./Preprocessed/Healthy/H(2).png");

        second.setIcon(imageIcon8)

        ImageIcon imageIcon9 =
new ImageIcon("./Preprocessed/Healthy/H(3).png");

        third.setIcon(imageIcon9);
    } catch (Exception e) {

        System.err.println("Error loading image: " + e.getMessage());

        e.printStackTrace();

    }

    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt)
{

        try {

            ImageIcon imageIcon4 =
new ImageIcon("./Preprocessed/Stone/S(1).png");

            first.setIcon(imageIcon4);

            ImageIcon imageIcon5 =
new ImageIcon("./Preprocessed/Stone/S(2).png");

            second.setIcon(imageIcon5);

```

```

        ImageIcon imageIcon6 =
new ImageIcon("./Preprocessed/Stone/S(3).png");

        third.setIcon(imageIcon6);
    } catch (Exception e) {

        System.err.println("Error loading image: " + e.getMessage());

        e.printStackTrace();

    }

}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt)
{

    // TODO add your handling code here:

    try {

        // Process files from Healthy directory

        File destDir = new File(Preprocessed_HealthyPath);

        if (!destDir.exists()) {

            destDir.mkdirs();

        }

        File[] healthyFiles = new File(Healthy_path).listFiles();

        if (healthyFiles != null) {

            for (File file : healthyFiles) {

                if (file.isFile()) {

                    resizeAndMoveImage(file, Preprocessed_HealthyPath);

```



```

        }

    }

}

// Process files from Stone directory

File destDir1 = new File(Preprocessed_StonePath);

if (!destDir1.exists()) {

    destDir1.mkdirs();

}

File[] stoneFiles = new File(Stone_path).listFiles();

if (stoneFiles != null) {

    for (File file : stoneFiles) {

        if (file.isFile()) {

            resizeAndMoveImage(file, Preprocessed_StonePath);

        }

    }

}

JOptionPane.showMessageDialog(this, "Images Resized and Moved
Successfully!");

} catch (Exception e) {

    JOptionPane.showMessageDialog(this, e);

    System.out.println(e);

```

```

    }

    }

    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt)
    {

        // TODO add your handling code here:

        new Feature_Extraction().setVisible(true);

        this.dispose();

    }

    /**

    * @param args the command line arguments

    */

    public static void main(String args[]) {

        /* Set the Nimbus look and feel */

        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting
code (optional) ">

        /* If Nimbus (introduced in Java SE 6) is not available, stay with the
default look and feel.

        *Fordetailssee
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

        */

        try {

            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels\(\)) {

```

```

        if ("Nimbus".equals(info.getName())) {

            javax.swing.UIManager.setLookAndFeel(info.getClassName());

            break;

        }

    }

} catch (ClassNotFoundException ex) {

    java.util.logging.Logger.getLogger(Preprocessing.class.getName()).log(java.util
        .logging.Level.SEVERE, null, ex);

    } catch (InstantiationException ex) {

        java.util.logging.Logger.getLogger(Preprocessing.class.getName()).log(java.util
            .logging.Level.SEVERE, null, ex);

    } catch (IllegalAccessException ex) {

        java.util.logging.Logger.getLogger(Preprocessing.class.getName()).log(java.util
            .logging.Level.SEVERE, null, ex);

    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

        java.util.logging.Logger.getLogger(Preprocessing.class.getName()).log(java.util
            .logging.Level.SEVERE, null, ex);

    }

}

//</editor-fold>

//</editor-fold>

```

```

    /* Create and display the form */

    java.awt.EventQueue.invokeLater(new Runnable() {

        public void run() {

            new Preprocessing().setVisible(true);

        }

    });

}

// Variables declaration - do not modify

private javax.swing.JLabel first;

private javax.swing.JButton jButton2;

private javax.swing.JButton jButton3;

private javax.swing.JButton jButton4;

private javax.swing.JButton jButton5;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JPanel jPanel1;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JScrollPane jScrollPane2;

private javax.swing.JScrollPane jScrollPane3;

private javax.swing.JLabel second;

```

```

        private javax.swing.JLabel third;
    }

    import ij.process.*;

    import ij.gui.*;

    import ij.plugin.*;

    import ij.plugin.frame.*;

    import ij.io.FileInfo;

    import java.awt.*;

    import java.awt.image.*;

    public class CompositeImage extends ImagePlus {

        /** Display modes (note: TRANSPARENT mode has not yet been
        implemented) */

        public static final int COMPOSITE=1, COLOR=2, GRAYSCALE=3,
        TRANSPARENT=4;

        public static final int MAX_CHANNELS = 7;

        int[] rgbPixels;

        boolean newPixels;

        MemoryImageSource imageSource;

        Image awtImage;

        WritableRaster rgbRaster;

```

```

SampleModel rgbSampleModel;

BufferedImage rgbImage;

ColorModel rgbCM;

ImageProcessor[] cip;

Color[] colors = {Color.red, Color.green, Color.blue, Color.white,
Color.cyan, Color.magenta, Color.yellow};

LUT[] lut;

int currentChannel = -1;

int previousChannel;

int currentSlice = 1;

int currentFrame = 1;

boolean singleChannel;

boolean[] active = new boolean[MAX_CHANNELS];

int mode = COLOR;

int bitDepth;

double[] displayRanges;

byte[][] channelLuts;

boolean customLuts;

boolean syncChannels;

public CompositeImage(ImagePlus imp) {

    this(imp, COLOR);

```

```
}
```

```
public CompositeImage(ImagePlus imp, int mode) {  
    if (mode<COMPOSITE || mode>GRAYSCALE)  
        mode = COLOR;  
    this.mode = mode;  
    int channels = imp.getNChannels();  
    bitDepth = getBitDepth();  
    if (IJ.debugMode) IJ.log("CompositeImage: "+imp+" "+mode+"  
"+channels);  
    ImageStack stack2;  
    boolean isRGB = imp.getBitDepth()==24;  
    if (isRGB) {  
        if (imp.getImageStackSize()>1)  
            throw new IllegalArgumentException("RGB stacks not supported");  
        stack2 = getRGBStack(imp);  
    } else  
        stack2 = imp.getImageStack();  
    int stackSize = stack2.getSize();  
    if (channels==1 && isRGB)
```

```

        channels = 3;

        if (channels==1 && stackSize<=MAX_CHANNELS &&
!imp.dimensionsSet)

        channels = stackSize;

        if (channels<1 || (stackSize%channels)!=0)

        throw new IllegalArgumentException("stacksize not multiple of
channels");

        if (mode==COMPOSITE && channels>MAX_CHANNELS)

        this.mode = COLOR;

        compositeImage = true;

        int z = imp.getNSlices();

        int t = imp.getNFrames();

        if (channels==stackSize || channels*z*t!=stackSize)

        setDimensions(channels, stackSize/channels, 1);

        else

        setDimensions(channels, z, t);

        setStack(imp.getTitle(), stack2);

        setCalibration(imp.getCalibration());

```



```

        FileInfo fi = imp.getOriginalFileInfo();

        if (fi!=null) {

displayRanges = fi.displayRanges;

channelLuts = fi.channelLuts;

        }

        setFileInfo(fi);

        Object info = imp.getProperty("Info");

        if (info!=null)

setProperty("Info", imp.getProperty("Info"));

        if (mode==COMPOSITE) {

            for (int i=0; i<MAX_CHANNELS; i++)

active[i] = true;

        } else

active[0] = true;

        //if (!(channels==3&&stackSize==3))

            setRoi(imp.getRoi());

            setOverlay(imp.getOverlay());

            if (channels!=stackSize)

```

```
setOpenAsHyperStack(true);  
  
}
```

```
public Image getImage() {  
  
    if (img==null)  
  
        updateImage();  
  
    return img;  
  
}
```

```
public void updateChannelAndDraw() {  
  
    if (!customLuts) singleChannel = true;  
  
        updateAndDraw();  
  
}
```

```
public void updateAllChannelsAndDraw() {  
  
    if (mode!=COMPOSITE)
```

```
        updateChannelAndDraw();  
  
        else {
```

```
            syncChannels = true;
```

```

singleChannel = false;

        updateAndDraw();

    }

}

public ImageProcessor getChannelProcessor() {

    if (cip!=null && currentChannel!=-1)

        return cip[currentChannel];

        else

            return getProcessor();

    }

    synchronized void setup(int channels, ImageStack stack2) {

        if (stack2!=null && stack2.getSize()>0 && (stack2.getProcessor(1)
instanceof ColorProcessor)) { // RGB?

            cip = null;

            lut = null;

return;

            }

            setupLuts(channels);

            if (mode==COMPOSITE) {

                cip = new ImageProcessor[channels];

                for (int i=0; i<channels; ++i) {

```

```

cip[i] = stack2.getProcessor(i+1);
cip[i].setLut(lut[i]);

}

```

```

currentSlice = currentFrame = 1;

        }

}

```

```

void setupLuts(int channels) {

    if (lut==null || lut.length<channels) {

if (displayRanges!=null && channels!=displayRanges.length/2)

displayRanges = null;

if (displayRanges==null&&ip.getMin()==0.0&&ip.getMax()==0.0)

```

```

ip.resetMinAndMax();

```

```

        lut = new LUT[channels];

```

```

        LUT                                lut2                                =

```

```

channels>MAX_CHANNELS?createLutFromColor(Color.white):null;

```

```

        for (int i=0; i<channels; ++i) {

```

```

            if (channelLuts!=null && i<channelLuts.length) {

```

```

lut[i] = createLutFromBytes(channelLuts[i]);

customLuts = true;

        } else if (i<MAX_CHANNELS)

            lut[i] = createLutFromColor(colors[i]);

else

lut[i] = (LUT)lut2.clone();

if (displayRanges!=null) {

lut[i].min = displayRanges[i*2];

lut[i].max = displayRanges[i*2+1];

} else {

lut[i].min = ip.getMin();

lut[i].max = ip.getMax();

}

}

displayRanges = null;

```

```

        }

    }

    public void resetDisplayRanges() {

        int channels = getNChannels();

        if (lut==null)

            setupLuts(channels);

            ImageStack stack2 = getImageStack();

            if (lut==null || channels!=lut.length || channels>stack2.getSize() ||
channels>MAX_CHANNELS)

                return;

                for (int i=0; i<channels; ++i) {

                    ImageProcessor ip2 = stack2.getProcessor(i+1);

                    ip2.resetMinAndMax();

                    lut[i].min = ip2.getMin();

                    lut[i].max = ip2.getMax();

                }

    }

```

```

public void updateAndDraw() {

    updateImage();

    if (win!=null)

notifyListeners(UPDATED);

    draw();

}

```

```

public synchronized void updateImage() {

    int imageSize = width*height;

    int nChannels = getNChannels();

    int redValue, greenValue, blueValue;

    int ch = getChannel();

    //IJ.log("updateImage:    "+ch+"/"+nChannels+"    "+currentSlice+"
"+currentFrame);

    if (ch>nChannels) ch = nChannels;

    boolean newChannel = false;

    if (ch-1!=currentChannel) {

previousChannel = currentChannel;

```

```
currentChannel = ch-1;
```

```
newChannel = true;
```

```
}
```

```
ImageProcessor ip = getProcessor();
```

```
if (mode!=COMPOSITE) {
```

```
    if (newChannel) {
```

```
        setupLuts(nChannels);
```

```
        LUT cm = lut[currentChannel];
```

```
        if (mode==COLOR)
```

```
            ip.setColorModel(cm);
```

```
                if (!(cm.min==0.0&&cm.max==0.0))
```

```
                    ip.setMinAndMax(cm.min, cm.max);
```

```
                        if (!IJ.isMacro()) ContrastAdjuster.update();
```

```
                            for (int i=0; i<MAX_CHANNELS; i++)
```

```
                                active[i] = i==currentChannel?true:false;
```



```

Channels.updateChannels();

}

if (ip!=null)

img = ip.createImage();

return;

                                }

if (nChannels==1) {

cip = null;

rgbPixels = null;

awtImage = null

if (ip!=null)

img = ip.createImage();

return;

}

```

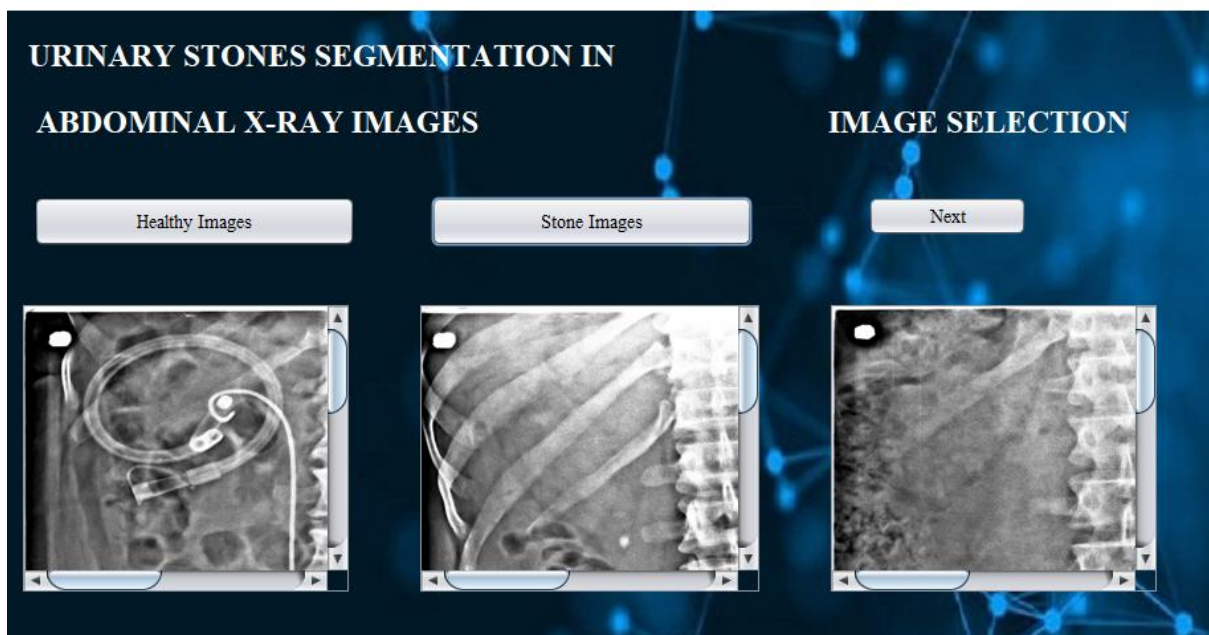
## APPENDIX II

### OUTPUT SCREENSHOTS

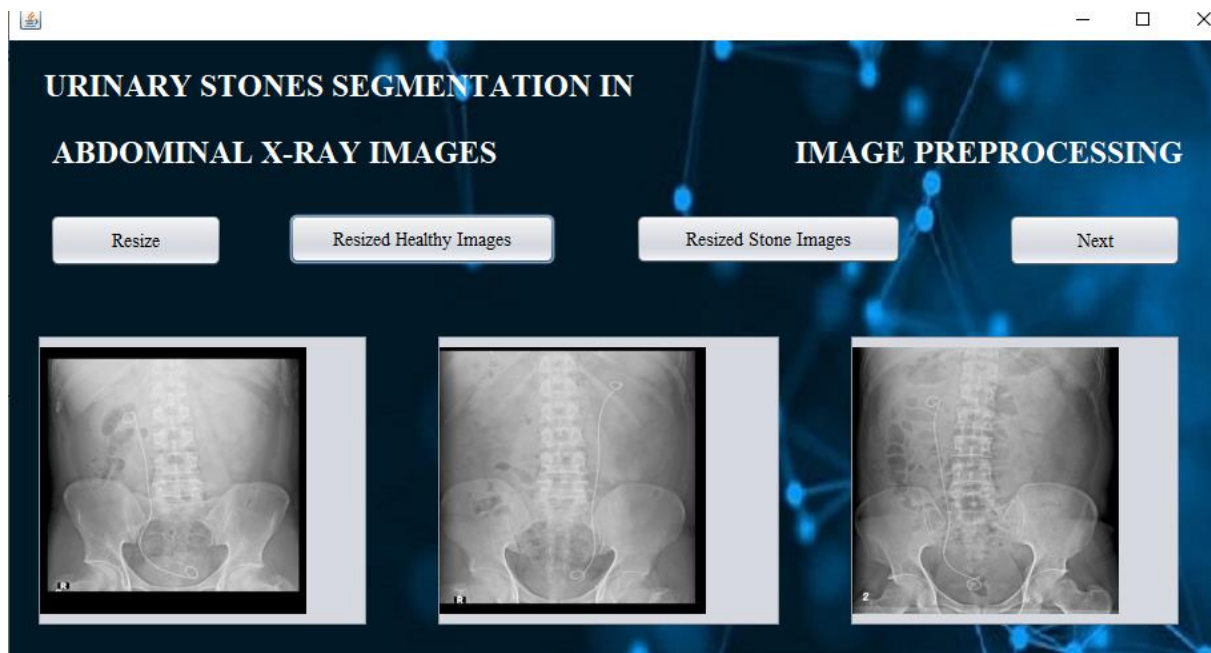
#### HOME PAGE



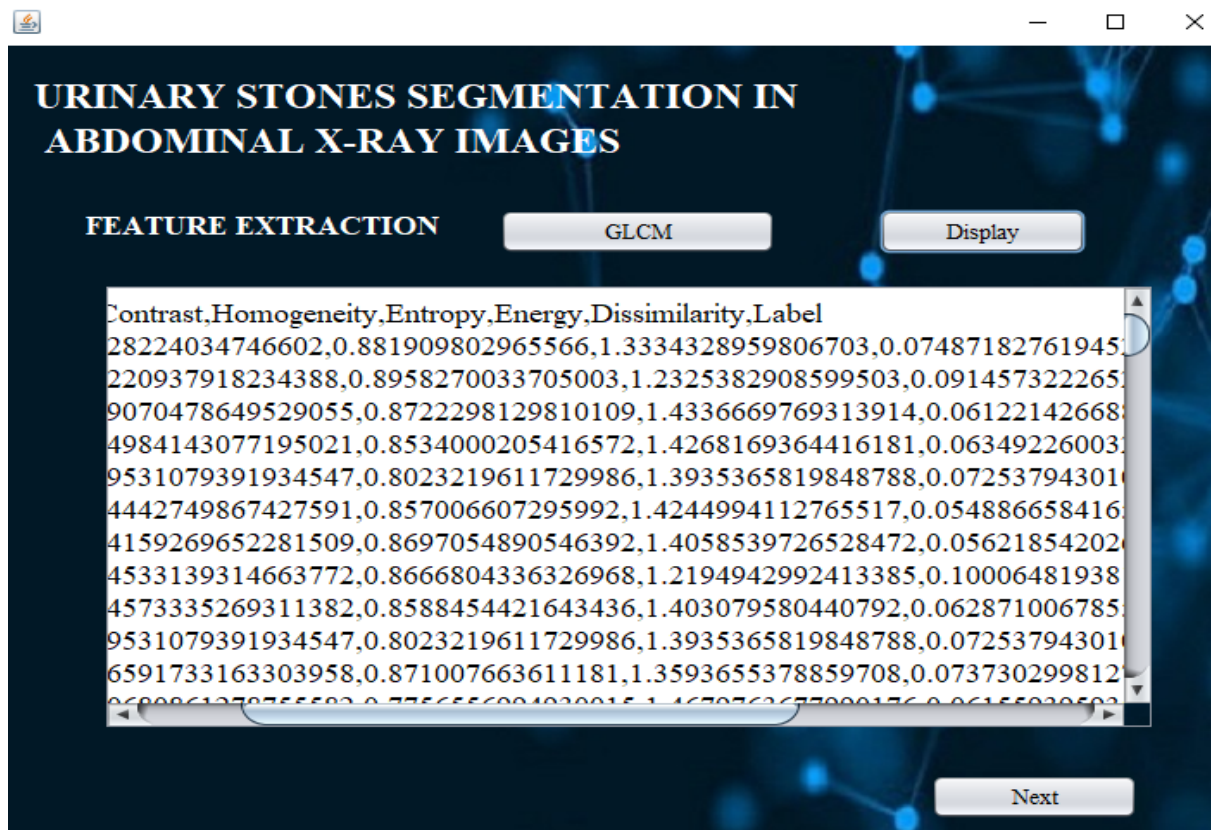
#### IMAGE SELECTION



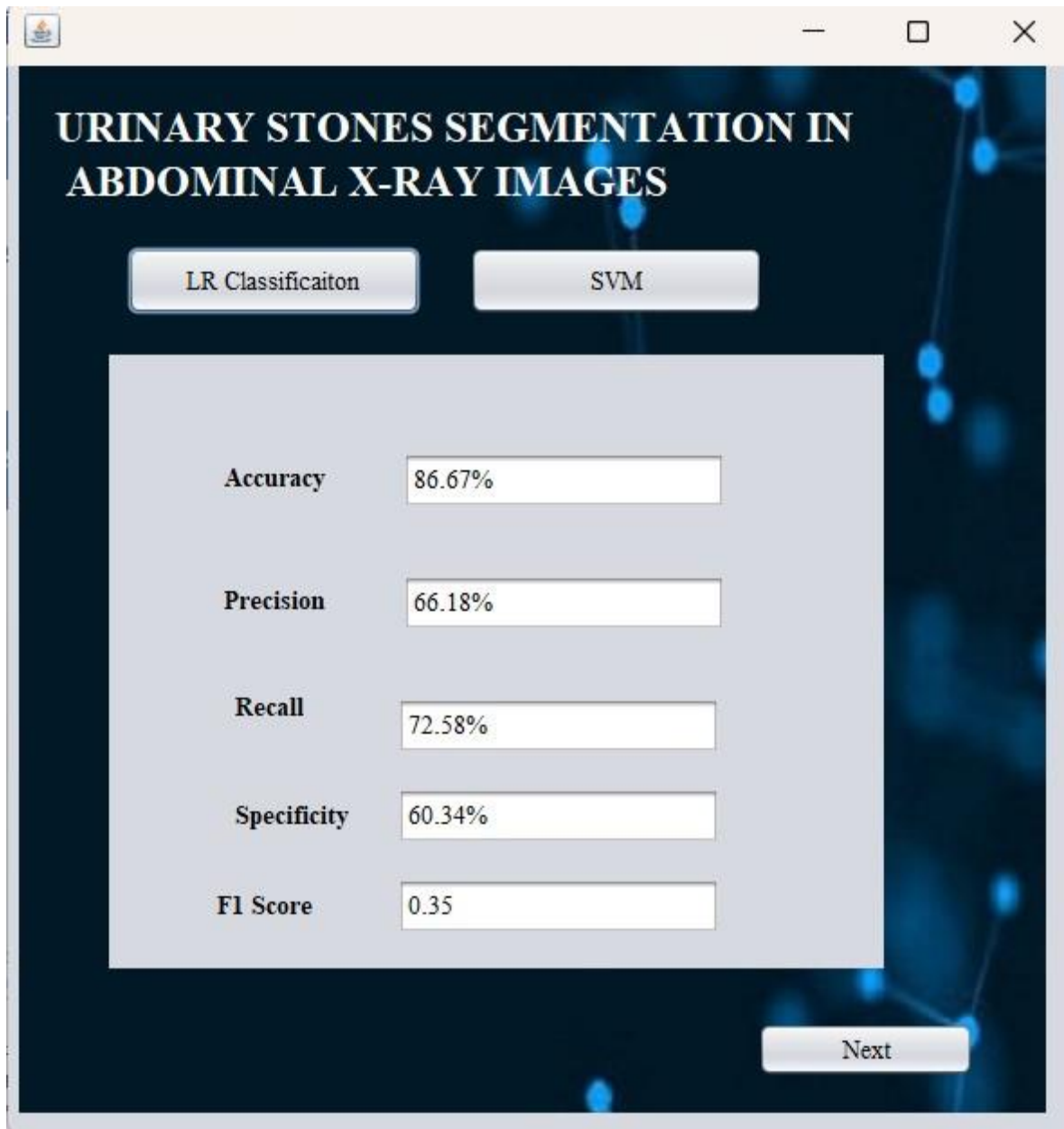
## PREPROCESSING



## FEATURE EXTRACTION:




## CLASSIFICATION:



The image shows a software window titled "URINARY STONES SEGMENTATION IN ABDOMINAL X-RAY IMAGES". Inside the window, there are two buttons at the top: "LR Classificaion" and "SVM". Below these buttons is a light gray rectangular area containing a table of classification metrics. The metrics are Accuracy (86.67%), Precision (66.18%), Recall (72.58%), Specificity (60.34%), and F1 Score (0.35). Each metric is displayed next to a text input field containing the value. At the bottom right of the window, there is a "Next" button.

Metric	Value
Accuracy	86.67%
Precision	66.18%
Recall	72.58%
Specificity	60.34%
F1 Score	0.35

— □ ×

# URINARY STONES SEGMENTATION IN ABDOMINAL X-RAY IMAGES

LR Classificaiton

SVM

Accuracy	90.56%
Precision	63.64%
Recall	94.38%
Specificity	47.25%
F1 Score	0.38

Next

## INPUT IMAGE:



## PREDICTION:

**URINARY STONES SEGMENTATION IN ABDOMINAL X-RAY IMAGES**

**Input Selection**

Browse

**Image Preprocessing**

Resize

**Feature Extraction**

GLCM

**Prediction**

Result

Predicted class: Stone

**Image Preprocessing Results**

Dissimilarity	0.3625856201782411
Homogeneity	0.8439736741998339
Entropy	1.3991037891304665
Energy	0.06863213914069712
Contrast	0.7269632847714753

**Size:** 1.0 mm

EXIT



## CHAPTER 11

### REFERENCES

- [1] Kenji Suzuki and Toshiaki kondo, “Urinary Stones Segmentation In Abdominal X-Ray Images Using Cascaded U-Net Pipeline With Stone Embedding Augmenataion And Lesion –Size Reweighting Approach” vol.11 Feb. 2023.
- [2] T. Alelign and B. Petros, “Kidney stone disease: An update on current concepts,” Adv. Urol., vol. 2018, pp. 1–12, Feb. 2018.
- [3] W. Brisbane, M. R. Bailey, and M. D. Sorensen, “An overview of kidney stoneimagingtechniques,” NatureRev.Urol.,vol.13,no.11,pp. 654–662, Nov. 2016, doi: 10.1038/nrurol.2016.154.
- [4] Tamilselvi, “Computer aided diagnosis system for stone detection and early detection of kidney stones,” J. Comput. Sci., vol. 7, no. 2, pp. 250–254, Feb. 2011, doi: 10.3844/jcssp.2011.250.254.
- [5] K. Viswanath and R. Gunasundari, “Design and analysis performance of kidney stone detection from ultrasound image by level set segmentation and ANN classification,” in Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI), Sep. 2014, pp. 407–414.
- [6] A. C. Patel and D. J. Frangos, "Shock wave lithotripsy: state of the art," Indian J. Urol., vol. 24, no. 2, pp. 155–160, Apr. 2008, doi: 10.4103/0970-1591.40609.
- [7] A. F. Aboumarzouk et al., "Shock wave lithotripsy (SWL) monotherapy for renal and ureteric stones: a systematic review of effectiveness and complications," Curr. Urol. Rep., vol. 14, no. 3, pp. 209–213, Jun. 2013, doi: 10.1007/s11934-013-0322-3.

- [8] P. C. Singh and G. C. Tater, "Comparison of ureteroscopic lithotripsy and extracorporeal shock wave lithotripsy in management of ureteric calculi: A prospective study," *J. Clin. Diagnostic Res.*, vol. 8, no. 8, pp. NC05-NC07, Aug. 2014, doi: 10.7860/JCDR/2014/9617.4768.
- [9] J. Ziemba, S. Jankowski, and L. Walentowicz-Sadlecka, "Dietary treatment of urinary risk factors for renal stone formation. A review of CLU Working Group," *Cent. European J. Urol.*, vol. 68, no. 1, pp. 100–106, 2015, doi: 10.5173/cej.2015.01.438.
- [10] G. Marchini, M. P. Monga, and T. R. Monga, "The contemporary metabolic syndrome rates in urolithiasis," *Adv. Chronic Kidney Dis.*, vol. 22, no. 5, pp. 404–411, Sep. 2015, doi: 10.1053/j.ackd.2015.05.009.
- [11] P. N. Shah, A. R. Bodiwala, and H. V. Jain, "A study on prevalence of urinary stones in Kheda district, Gujarat," *Natl. J. Community Med.*, vol. 8, no. 11, pp. 666–669, Nov. 2017.
- [12] J. E. Worcester and F. L. Coe, "Clinical practice. Calcium kidney stones," *New Engl. J. Med.*, vol. 363, no. 10, pp. 954–963, Sep. 2010, doi: 10.1056/NEJMc1001011.
- [13] S. K. Shaw, S. M. Sankhwar, and A. Saxena, "Recent trends in the epidemiology of urolithiasis in rural India," *Indian J. Urol.*, vol. 34, no. 3, pp. 214–219, Jul. 2018, doi: 10.4103/iju.IJU\_251\_17.
- [14] M. Ferraro, L. Palumbo, and G. G. Grimaldi, "New technologies for the treatment of urinary stones: shock waves and ureteroscopy," *Urologia*, vol. 83, no. 2, pp. 55–58, Apr. 2016, doi: 10.5301/uro.5000151.



- [15] K. L. Penniston and S. Nakada, "Medical management of kidney stones: AUA guideline," *J. Urol.*, vol. 192, no. 2, pp. 316–324, Aug. 2014, doi: 10.1016/j.juro.2014.05.006.
- [16] A. P. Wolff et al., "The association of dietary factors with subtypes of kidney stones in the EPIC-Greece cohort," *J. Urol.*, vol. 194, no. 4, pp. 895–901, Oct. 2015, doi: 10.1016/j.juro.2015.03.118.
- [17] S. K. Singh et al., "Prevalence of kidney stones and associated risk factors in the Northern region of Bangladesh: a cross-sectional study," *Clin. Epidemiol. Glob. Health*, vol. 8, no. 3, pp. 938–942, Jul. 2020, doi: 10.1016/j.cegh.2019.10.007.
- [18] J. C. Lieske, "Recent advances in understanding and management of nephrolithiasis: impact of case-based learning on medical management," *Kidney Res. Clin. Pract.*, vol. 38, no. 3, pp. 273–278, Sep. 2019, doi: 10.23876/j.krcp.19.016.
- [19] L. Ferraro, G. G. Grimaldi, and L. Palumbo, "Ultrasound imaging and contrast-enhanced ultrasound in urinary stone disease," *Urologia*, vol. 84, no. 2, pp. 57–62, May 2017, doi: 10.5301/uro.5000205.
- [20] M. M. Thiam, B. A. Faye, and S. M. Fall, "Epidemiological and clinical aspects of urolithiasis in Dakar: a hospital-based study," *Prog. Urol.*, vol. 28, no. 1, pp. 31–37, Jan. 2018, doi: 10.1016/j.purol.2017.11.009.
- [21] G. Marchini et al., "Stone disease in the era of the metabolic syndrome: an update," *Int. Braz. J. Urol.*, vol. 44, no. 4, pp. 799–808, Jul-Aug 2018, doi: 10.1590/S1677-5538.IBJU.2017.0417.

- [22] T. Shafi, N. S. Zafar, and N. A. Ehsan, "Demographic profile of urinary stone patients in a tertiary care hospital of Lahore, Pakistan," *Pakistan J. Med. Sci.*, vol. 35, no. 3, pp. 667–671, May-Jun 2019, doi: 10.12669/pjms.35.3.209.
- [23] J. W. Rule et al., "Temporal changes in kidney stone composition and in risk factors predisposing to stone formation," *J. Urol.*, vol. 195, no. 1, pp. 230–235, Jan. 2016, doi: 10.1016/j.juro.2015.08.088.
- [24] A. G. Oyedele et al., "Prevalence and predictors of urinary stone disease among patients with prostate cancer on androgen deprivation therapy," *Indian J. Urol.*, vol. 33, no. 1, pp. 43–48, Jan-Mar 2017, doi: 10.4103/0970-1591.194769.
- [25] D. Sivalingam et al., "Ultrasound in the evaluation of renal colic: is it really helpful?," *Int. J. Emerg. Med.*, vol. 12, no. 1, p. 11, Mar. 2019, doi: 10.1186/s12245-019-0225-6.