

# Ai Fracture Detection and Smart Hospital Queue Management System: A Research Paper

**Aswin S Sajeev**

Good Shepherd

College of Engineering and Technology

**Jenisha P**

Good Shepherd

College of Engineering and Technology

**Jino N J**

Good Shepherd

College of Engineering and Technology

**Saranya S**

Good Shepherd

College of Engineering and Technology

## ABSTRACT

The increasing demand for efficient healthcare services has encouraged the integration of intelligent technologies to assist medical diagnosis and hospital management. This paper presents an AI-Based Fracture Detection and Smart Hospital Queue Management System, developed using Python and image processing techniques. The system aims to assist in detecting bone fractures from X-ray images while improving hospital queue efficiency.

The fracture detection module uses grayscale conversion, edge detection, and contour analysis to identify fracture patterns. Based on these patterns, the system calculates fracture severity and estimates recovery time. The queue management module generates tokens, maintains real-time status, and notifies patients.

The system is implemented using Streamlit, providing an interactive interface. Results show improved efficiency in both diagnosis support and patient management. This system demonstrates how AI and automation can enhance healthcare services.

## KEYWORDS

Artificial Intelligence, Fracture Detection, Image Processing, X-ray Analysis, Queue Management, Streamlit.

## 1. INTRODUCTION

Healthcare systems face challenges in both diagnosis and patient management. Traditional fracture detection relies on manual analysis, which can be slow and error-prone. Similarly, hospital queue systems often result in long waiting times and poor organization.

This project introduces a system combining fracture detection and queue management.

## 2. LITERATURE REVIEW

The integration of artificial intelligence and image processing in healthcare has significantly improved diagnostic accuracy and hospital management systems. This section reviews existing research and technologies related to fracture detection and smart queue management systems.

## 2.1 RELATED WORKS

### 2.1.1 AI in Medical Image Analysis

Medical image analysis has become a major application area of artificial intelligence. Researchers have extensively used **deep learning techniques**, especially Convolutional Neural Networks (CNNs), for detecting fractures in X-ray images. These models are trained on large datasets and can automatically identify patterns and abnormalities with high accuracy.

Studies show that CNN-based models outperform traditional methods in fracture detection tasks. However, these models require high computational power, large labeled datasets, and longer training time. This makes them less suitable for small-scale or real-time applications.

In contrast, traditional image processing techniques such as grayscale conversion, edge detection, and contour analysis are computationally efficient and easier to implement. These methods are widely used in systems where quick processing and low resource usage are required. Although their accuracy may be lower than deep learning approaches, they provide a reliable and fast solution for preliminary diagnosis.

### 2.1.2 Smart Healthcare Management Systems

Modern hospitals are adopting digital solutions to improve patient flow and reduce waiting times. Various queue management systems have been developed to automate patient handling using token systems, scheduling algorithms, and real-time displays.

Some advanced systems use machine learning to predict patient waiting times and optimize scheduling. Others integrate IoT devices and mobile applications to provide real-time updates and notifications to patients. While these systems improve efficiency, they often require complex infrastructure and are expensive to implement.

Simple queue systems based on token generation are still widely used due to their ease of implementation and reliability. However, many of these systems lack features such as real-time notifications, remote access, and integration with diagnostic modules.

### 2.1.3 Image Processing Techniques for Fracture Detection

Image processing plays a crucial role in detecting fractures from X-ray images. Techniques such as **grayscale conversion** simplify image data by reducing color complexity, making it easier to analyze structural details.

**Edge detection algorithms**, particularly the Canny Edge Detection method, are widely used to identify boundaries in medical images. This helps in highlighting fracture lines and discontinuities in bone structures.

**Contour analysis** is another important technique used to detect irregular shapes and patterns in images. By analyzing contours, the system can identify deviations in normal bone structure, which may indicate fractures.

These techniques are effective for identifying visible fractures and are computationally efficient. However, their performance depends on image quality and may not detect very subtle fractures.

## 2.2 RESEARCH GAPS

Despite advancements in both medical image analysis and hospital management systems, several limitations exist in current solutions:

- Most fracture detection systems rely on deep learning models, which require high computational resources
- Traditional systems lack integration between diagnosis and hospital management
- Existing queue systems do not provide real-time notifications or smart interaction
- Many solutions are not cost-effective for small or medium healthcare facilities
- Limited availability of systems that combine both medical analysis and patient management

## 2.3 PROPOSED APPROACH CONTRIBUTION

The proposed system addresses these gaps by combining **image processing-based fracture detection** with a **smart queue management system** in a single platform. Unlike complex deep learning models, the system uses lightweight algorithms that are efficient and easy to implement.

The integration of these two modules provides a unique solution that improves both diagnostic support and hospital workflow. The system is designed to be cost-effective, user-friendly, and suitable for real-time applications.

## 3. METHODOLOGY

The methodology of the proposed **AI-Based Fracture Detection and Smart Hospital Queue Management System** describes the systematic approach used to design, develop, and implement the system. The system integrates two major modules: **fracture detection using image processing** and **hospital queue management using automated token handling**. Both modules are designed to work independently while being integrated within a single application.

### 3.1 SYSTEM ARCHITECTURE

The system follows a modular and layered architecture to ensure efficient data flow and easy maintenance. It consists of three main layers:

- **Input Layer:** Accepts user inputs such as X-ray images (upload or camera) and patient details.
- **Processing Layer:** Performs image analysis and queue management operations.
- **Output Layer:** Displays results such as fracture severity, recovery time, and queue status.

This architecture ensures that each module operates independently, improving system scalability and performance.

## 3.2 FRACTURE DETECTION METHODOLOGY

The fracture detection process is based on image processing techniques and follows a step-by-step pipeline:

### 3.2.1 Image Acquisition

The system accepts X-ray images either through file upload (JPG, PNG formats) or real-time camera input. This flexibility allows the system to work in both offline and real-time scenarios.

### 3.2.2 Image Preprocessing

The input image is converted into grayscale format to reduce complexity and focus on intensity variations. Preprocessing may also include noise reduction to improve image quality.

### 3.2.3 Edge Detection

The **Canny Edge Detection algorithm** is applied to detect edges in the image. This step highlights boundaries and possible fracture lines by identifying sharp intensity changes.

### 3.2.4 Contour Detection

Contours are extracted from the edge-detected image using OpenCV functions. These contours represent object boundaries and help identify irregularities in bone structures.

### 3.2.5 Fracture Severity Estimation

The number of detected contours is used as a metric to estimate fracture severity. A higher number of contours may indicate more complex fracture patterns.

### Mathematical Representation

Let:

F= Fracture Severity Percentage

C= Number of Contour

$$F = \min(C, 100)$$

### 3.2.6 Classification and Recovery Estimation

- Mid:  $F < 30$
- Moderate:  $30 \leq F < 70$
- Severe:  $F \geq 70$

Recovery time is estimated as:

Severity =	Mid	$F < 30$
	Moderate	$30 \leq F < 70$
	Sever	$F \geq 70$

### 3.3 HOSPITAL QUEUE MANAGEMENT METHODOLOGY

The queue management system is designed to handle patient flow efficiently using a token-based approach.

#### 3.3.1 Patient Registration

The system collects patient details such as name, phone number, department, and doctor information.

#### 3.3.2 Token Generation

Each patient is assigned a unique token number generated sequentially.

$$T_n = T_{n-1} + 1$$

Where:

$T_n$  = Current token number

#### 3.3.3 Queue Storage

Patient details are stored using session state, allowing real-time updates within the application.

#### 3.3.4 Queue Display

The queue is displayed in tabular format using Pandas, providing clear visibility of patient status.

#### 3.3.5 Patient Calling Mechanism

The system calls the next patient based on the queue order and updates their status from “Waiting” to “Called”.

#### 3.3.6 Notification System

A reminder notification is sent to the next patient, informing them that their turn is approaching.

### 3.4 SYSTEM FLOW

The overall workflow of the system is as follows:

1. User selects module (Fracture Detection / Queue Management)
2. Input is provided (image or patient details)
3. Data is processed using appropriate algorithms

4. Results are generated (fracture analysis or queue status)
5. Output is displayed on the user interface

## 4. IMPLEMENTATION

The implementation of the proposed **AI-Based Fracture Detection and Smart Hospital Queue Management System** was carried out using Python and modern web-based tools. The system was developed as an interactive application to ensure ease of use, real-time processing, and efficient performance. This section describes the development environment, system modules, and implementation details of each component.

### 4.1 DEVELOPMENT ENVIRONMENT

The system was developed using the following technologies:

- **Programming Language:** Python
- **Frontend Framework:** Streamlit
- **Image Processing Library:** OpenCV
- **Numerical Computation:** NumPy
- **Data Handling:** Pandas

The development was carried out in a local environment, and the application runs on a web browser using Streamlit interface. This ensures cross-platform compatibility and easy deployment.

### 4.2 FRACTURE DETECTION MODULE IMPLEMENTATION

The fracture detection module was implemented using image processing techniques provided by OpenCV. The module accepts X-ray images as input and processes them through multiple stages.

Initially, the input image is converted into a NumPy array for processing. If the image is in grayscale format, it is converted into a three-channel format to maintain consistency. The image is then converted back to grayscale to simplify further processing.

The **Canny Edge Detection algorithm** is applied to detect edges within the image. This step highlights the boundaries of bones and potential fracture lines. After edge detection, contour detection is performed using OpenCV functions. The number of contours detected is used as a measure to estimate fracture severity.

The results are displayed using Streamlit components such as:

- Image display
- Severity percentage (metric)

- Progress bar visualization
- Recovery time estimation

This module provides a clear and interactive way to analyze X-ray images.

### 4.3 IMAGE INPUT AND CAMERA PROCESSING

The system supports two input methods:

#### 1. **Image Upload:**

Users can upload X-ray images in formats such as JPG, JPEG, and PNG. The uploaded image is processed immediately after selection.

#### 2. **Real-Time Camera Input:**

The system uses OpenCV to access the device camera and capture frames in real time. The captured frames are displayed dynamically on the interface.

The camera module runs for a limited number of frames to ensure system stability and avoid excessive resource usage.

### 4.4 HOSPITAL QUEUE MANAGEMENT IMPLEMENTATION

The hospital queue management system is implemented using Python data structures and Streamlit session state. When the application starts, session variables such as queue list, token counter, and current token are initialized.

During patient registration, user inputs are collected through form fields. Once submitted, a unique token number is generated, and the patient details are stored in the queue. The token counter is incremented automatically.

The queue is displayed in tabular format using Pandas Data Frame, allowing easy visualization of patient details such as:

- Token number
- Patient name
- Department
- Doctor
- Status (Waiting/Called)

The system dynamically updates the queue whenever a new patient is added or when a patient is called.

## 4.5 PATIENT TOKEN AND NOTIFICATION SYSTEM

The token system ensures that patients are served in an organized manner. When the administrator selects “Call Next Patient,” the system identifies the first patient with “Waiting” status and updates it to “Called.”

A notification feature is implemented using Streamlit’s alert system. When a patient is called, the next patient in the queue receives a reminder message indicating that their turn is approaching. This improves efficiency and reduces waiting time.

## 4.6 USER INTERFACE IMPLEMENTATION

The user interface is developed using Streamlit, which provides an easy-to-use web interface. The layout includes:

- Sidebar navigation for module selection
- Input fields for image upload and patient details
- Buttons for actions such as analysis and token generation
- Output sections for displaying results and queue status

The interface is designed to be simple, interactive, and user-friendly, allowing both technical and non-technical users to operate the system easily.

## 4.7 TESTING AND DEBUGGING

The system was tested during development to ensure correct functionality. Various test cases were used, including different X-ray images and patient data inputs.

Errors related to image processing, data handling, and UI interaction were identified and corrected. Debugging was performed using Python tools and manual testing methods to ensure system stability.

## 5. RESULTS AND DISCUSSION

The proposed **AI-Based Fracture Detection and Smart Hospital Queue Management System** was tested under various conditions to evaluate its performance and usability. The results indicate that the system effectively achieves its intended objectives by combining image processing with real-time queue management.

### 5.1 SYSTEM RESULTS

The system successfully performs the following functions:

- Detects fracture patterns from X-ray images
- Estimates fracture severity using contour analysis
- Classifies fractures into mild, moderate, and severe levels

- Provides approximate recovery time
- Manages patient queue efficiently using token generation
- Displays real-time queue status
- Reduces waiting time through organized patient flow

The outputs are displayed in an interactive format using the Streamlit interface, making them easy to understand.

## 5.2 FRACTURE DETECTION RESULTS

The fracture detection module was tested with different X-ray images. The system was able to detect edges and contours effectively, highlighting possible fracture regions.

Key observations include:

- Clear images produced more accurate results
- Edge detection successfully identified bone boundaries
- Contour count provided a reasonable estimation of severity

The system provides a useful preliminary analysis, although it is not a replacement for professional medical diagnosis.

## 5.3 QUEUE MANAGEMENT RESULTS

The queue management system was tested with multiple patient entries. The system performed reliably by:

- Generating unique tokens for each patient
- Maintaining proper queue order
- Updating patient status dynamically
- Providing notifications to upcoming patients

This helped reduce confusion and improved the overall patient flow.

## 5.4 PERFORMANCE ANALYSIS

The system performance was evaluated based on:

- **Speed:** Fast image processing and real-time updates
- **Usability:** Simple and user-friendly interface
- **Efficiency:** Runs smoothly on standard systems
- **Accuracy:** Provides consistent results for visible fractures

## 5.5 DISCUSSION

The system demonstrates that integrating image processing with queue management can improve healthcare efficiency. While the fracture detection module provides quick analysis, its accuracy depends on image quality. The queue system is effective for small-scale environments but may require enhancements for large hospitals.

## REFERENCES

The following references were used during the development and study of the project:

1. Gonzalez, R. C., & Woods, R. E. (2018). Digital Image Processing (4th Edition). Pearson Education.
2. Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly Media.
3. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
4. Python Software Foundation. (2024). Python Documentation. Available at: <https://www.python.org>
5. OpenCV Documentation. (2024). Open Source Computer Vision Library. Available at: <https://opencv.org>
6. McKinney, W. (2017). Python for Data Analysis: Data Wrangling with Pandas and NumPy. O'Reilly Media.
7. Streamlit Documentation. (2024). Streamlit Web Application Framework. Available at: <https://streamlit.io>
8. Jain, A. K. (2019). Fundamentals of Digital Image Processing. Prentice Hall.
9. Szeliski, R. (2022). Computer Vision: Algorithms and Applications. Springer.
10. Healthcare Information and Management Systems Society (HIMSS). (2023). Smart Healthcare Technologies and Applications.