# VISVESVARAYA TECHNOLOGICAL UNIVERSITY,

# BELAGAVI - 590018

## Mini Project Report

## On

# "DETECTION OF FRACTURE IN HAND USING IMAGE PROCESSING"

**A report submitted in partial fulfillment of the requirements for**

**COMPUTER GRAPHICS AND IMAGE PROCESSING LABORATORY (21CSL66)**

**In**

**Computer Science and Design**

**Submitted by**


| | |
|---|---|
| **MANOJ M** | **4AL21CG036** |
| **SHARVARI M S** | **4AL21CG049** |

**Under the Guidance of**

**Dr. Pushparani M K**

**Senior Assistant Professor**

**DEPARTMENT OF COMPUTER SCIENCE AND DESIGN**

**ALVA'S INSTITUTE OF ENGINEERING & TECHNOLOGY MIJAR,**

(Unit of Alva's Education Foundation ®, Moodbidri)

Affiliated to Visvesvaraya Technological University, Belagavi,

Approved by AICTE, New Delhi, Recognized by the Government of Karnataka.

**Accredited by NACC with A+ Grade**

Shobavana Campus, Mijar, Moodbidri, D.K., Karnataka 2023-2024

# ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY MIJAR, MOODBIDRI, D.K. -574225


## DEPARTMENT OF COMPUTER SCIENCE AND DESIGN

## CERTIFICATE

This is to certify that the Computer Graphics and Image Processing Laboratory with Mini Project entitled **"DETECTION OF FRACTURE IN HAND USING IMAGE PROCESSING"** has been completed by

| | |
|---|---|
| MANOJ M | 4AL21CG036 |
| SHARVARI M S | 4AL21CG049 |

The Bonafide students of the **Department of Computer Science & Design, Alva's Institute of Engineering and Technology** in **DEPARTMENT OF COMPUTER SCIENCE & DESIGN** of the **VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI** during the year 2023–2024. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The Mini Project report has been approved as it satisfies the academic requirements concerning the Mini Project work of Computer Graphics and Image Processing subject prescribed for the Bachelor of Engineering Degree.

**Dr. Pushparani M K**
**Mini Project Guide**

**Prof. Jayantkumar A Rathod**
**HOD, Dept. of CSD**

# ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY MIJAR, MOODBIDRI D.K. -574225


## DEPARTMENT OF COMPUTER SCIENCE AND DESIGN

## DECLARATION

We,

**MANOJ M**

**SHARVARI M S**

Here by declare that the dissertation entitled, **DETECTION OF FRACTURE IN HAND USING IMAGE PROCESSING** is completed and written by us under the supervision of our guide Dr. **Pushparani M K**, **Senior Assistant Professor,** Department of Computer Science and Design Alvas's Institute of Engineering and Technology, Moodbidri, **DEPARTMENT OF COMPUTER SCIENCE AND DESIGN** of the **VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI** during the academic year 2023-2024. The dissertation report is original and it has not been submitted for any other degree in any university.

| | |
|---|---|
| MANOJ M | 4AL21CG036 |
| SHARVARI M S | 4AL21CG049 |

# ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany a successful completion of any task would be incomplete without the mention of the people who made it possible, success is the epitome of handwork and perseverance, but steadfast of all is encouraging guidance.

So, with gratitude, we acknowledge all those whose guidance and encouragement served as a beacon of light and crowned the effort with success.

The selection of this mini-project work as well as the timely completion is mainly due to the interest and persuasion of our mini-project guide **Dr. Pushparani M K,** Senior Assistant Professor, Department of Computer Science and Design. We will remember her contribution forever.

We sincerely thank, **Prof. Jayanthkumar A Rathod**, Head of the Department of Computer Science and Design who has been the constant driving force behind the completion of the project.

We thank our beloved Principal, **Dr. Peter Fernandes,** for his constant help and support throughout.

We are indebted to the **Management of Alva's Institute of Engineering and Technology, Mijar, Moodbidri** for providing an environment that helped us in completing our mini project.

Also, we thank all the teaching and non-teaching staff of the Department of Computer Science and Design for the help rendered.

MANOJ M 4AL21CG036

SHARVARI M S 4AL21CG049

# ABSTRACT

The detection of fractures in hand X-ray images is a critical task in the medical field, often requiring expert radiologists to perform detailed analyses. This project aims to develop an automated system for detecting fractures using image processing techniques and machine learning algorithms. The proposed system enhances X-ray images, extracts relevant features, and applies machine learning models to identify potential fractures. By automating the detection process, the system assists medical professionals in making quicker and more accurate diagnoses, ultimately improving patient outcomes.

The implementation leverages Python programming along with libraries such as OpenCV, PIL, and TensorFlow. The system architecture includes components for image acquisition, pre-processing, feature extraction, fracture detection, and result visualization. Techniques such as histogram equalization, noise reduction, edge detection, and image segmentation are employed to process the images effectively.

The project demonstrates high accuracy in detecting fractures when tested with a dataset of hand X-ray images, showing significant potential for reducing diagnostic time and improving consistency. Future enhancements could include integration with hospital information systems, expansion to other types of fractures, and the application of advanced deep learning techniques to further improve detection performance.

In conclusion, this project provides a robust and efficient solution for fracture detection in hand X-ray images, showcasing the power of image processing and machine learning in medical diagnostics.

# TABLE OF CONTENTS

# INTRODUCTION

In recent years, machine learning and image processing advancements have revolutionized many fields, including healthcare. Medical imaging, in particular, has seen significant improvements in diagnostic accuracy and efficiency due to these technologies. Hand fractures, a common type of injury, require precise and swift diagnosis to prevent complications and ensure proper healing. Traditionally, this diagnosis is made by radiologists who examine X-ray images. However, this manual process can be prone to errors and delays. According to studies, misdiagnosis or delayed diagnosis of fractures can lead to improper treatment, prolonged recovery times, and in severe cases, permanent disability.

The development of automated systems for fracture detection can alleviate these issues by providing quick, reliable, and consistent results. This project leverages the YOLO model, a state-of-the-art object detection algorithm, to identify hand fractures from X-ray images. The YOLO model is known for its speed and accuracy, making it an ideal choice for real-time applications. By integrating this model with a graphical user interface, we aim to create a tool that can assist healthcare professionals in diagnosing hand fractures more efficiently. The system not only identifies fractures but also offers precautionary guidelines based on the type of fracture detected, helping patients understand the immediate steps they should take.

The following sections will provide a detailed overview of the project's methodology, implementation, reviews, results, and conclusions.

## 1.1 PROBLEM STATEMENT

**Manual Inspection**: The reliance on manual inspection by radiologists can lead to delays and potential oversight due to human error.

**Diagnostic Consistency**: There is a need for a consistent diagnostic tool that can provide reliable results across different cases.

**Efficiency**: Improving the efficiency of fracture detection can lead to quicker diagnoses and treatment, benefiting patient outcomes.

## 1.2 WHAT IS YOLO

The YOLO (You Only Look Once) model is a state-of-the-art, real-time object detection system developed by Joseph Redmon, Ali Farhadi, and Santosh Divvala. Unlike traditional object detection methods that apply a model to an image at multiple locations and scales, YOLO frames object detection as a single regression problem, directly predicting bounding boxes and class probabilities from full images in one evaluation. This innovative approach allows YOLO to be incredibly fast and efficient, making it suitable for real-time applications.

- YOLO divides the input image into a grid of SxS cells. Each cell is responsible for predicting a certain number of bounding boxes and their confidence scores.
- The confidence score reflects the likelihood that a bounding box contains an object and the accuracy of the bounding box's position.
- Each cell also predicts class probabilities for the detected objects, assuming the object center falls within the cell.
- Unlike region proposal-based systems (e.g., Faster R-CNN) that first generate potential object regions and then classify them, YOLO processes the entire image in one forward pass through the neural network.
- This architecture significantly reduces computation time and enables real-time detection capabilities.
- Each grid cell predicts multiple bounding boxes, each with its confidence score and class probabilities.
- YOLO's grid-based prediction approach helps in localizing objects and assigning appropriate class labels simultaneously.

## 1.3 WHAT IS GUI DEVELOPMENT

GUI (Graphical User Interface) development involves creating interfaces that enable users to interact with software applications through graphical elements like buttons, menus, text fields, and icons, instead of text-based commands. This type of development focuses on enhancing user experience (UX) by making software more intuitive and accessible.

Key concepts in GUI development include the layout and design of user interface (UI) components, which comprise the visual elements that users interact with, and event-driven programming, where the application responds to user actions such as clicks and keystrokes. Several tools and libraries facilitate GUI development across different programming languages.

For instance, Tkinter is a standard GUI library for Python, providing a straightforward way to create desktop applications. PyQt and Kivy are other Python libraries offering more advanced features and support for multitouch applications, respectively. Java developers might use JavaFX or Swing for rich internet applications and customizable components. Electron, using web technologies like HTML, CSS, and JavaScript, is popular for building cross-platform desktop apps. The process of GUI development generally involves planning and designing the interface, selecting the appropriate tools, building the UI components, implementing event handling to manage user interactions, testing and debugging the application, and refining the interface based on feedback and testing results. By focusing on usability and accessibility, GUI development aims to create software that is not only functional but also user-friendly and visually appealing.

# 2.METHODOLOGY

The methodology for detecting hand fractures using image processing involves several key steps:

## 2.1 Data Collection and Preparation:

- Collecting a dataset of hand X-ray images that include various types of fractures. This involves sourcing images from publicly available medical databases and collaborating with medical institutions.
- Annotating the images with bounding boxes to indicate the locations of fractures. This step is crucial as it provides the ground truth data needed for training the model.
- Splitting the dataset into training and testing sets to train and evaluate the model. Typically, 80% of the data is used for training, and 20% is reserved for testing to assess the model's performance.

## 2.2 Model Selection and Training:

- Selecting the YOLO model due to its proven effectiveness in object detection tasks. YOLO's architecture allows it to process images quickly, making it suitable for real-time applications.
- Training the YOLO model on the prepared dataset, fine-tuning it to recognize different types of hand fractures. The training process involves multiple iterations, adjusting hyperparameters such as learning rate and batch size to optimize performance.
- Evaluating the model's performance using metrics such as precision, recall, and F1-score to ensure its accuracy. Cross-validation techniques may also be employed to prevent overfitting and ensure the model generalizes well to unseen data.

## 2.3 GUI Development:

- Developing a graphical user interface using the Tkinter library in Python. The GUI is designed to be intuitive, allowing users to upload X-ray images easily and view detection results.
- Implementing features for image upload, fracture detection, and display of precautionary guidelines. The interface includes buttons for these actions and areas where results and guidelines are displayed clearly.
- Ensuring the GUI is user-friendly and accessible to medical professionals. Feedback from potential users is incorporated to refine the interface, making it suitable for use in clinical settings.

## 2.4 Integration and Testing:

- Integrating the trained YOLO model with the GUI to enable real-time fracture detection. This involves linking the model's output with the display components of the GUI.
- Conducting thorough testing with various X-ray images to validate the system's performance. This step ensures the system works reliably under different conditions and with different types of fractures.

- Gathering feedback from potential users to refine and improve the system. Medical professionals' input helps identify any usability issues and ensures the system meets their needs effectively.

# 3.IMPLEMENTATION

## 3.1 Libraries and Modules

These libraries and modules are integral to the development of the hand fracture detection system, providing essential functionalities that enable image processing, GUI creation, and object detection.

- **tkinter:**
  - **Purpose:** Used for creating the graphical user interface (GUI).
  - **Details:** Tkinter is a standard GUI library in Python, known for its simplicity and ease of use. It provides a robust framework to develop desktop applications with various widgets like buttons, labels, text fields, and more. In this project, Tkinter is used to create the main application window, buttons for image upload, and areas to display results. Its event-driven architecture allows for interactive user interfaces where specific functions are triggered by user actions, such as button clicks.
- **filedialog:**
  - **Purpose:** Facilitates the selection of image files for upload.
  - **Details:** Filedialog is a module in Tkinter that allows users to open a file dialog box to browse and select files from their file system. This is particularly useful for uploading X-ray images, as it provides a familiar and straightforward interface for users to navigate their directories and select the required image. This module ensures that users can easily upload images without needing to manually enter file paths.
- **PIL (Pillow):**
  - **Purpose:** Handles image display within the GUI.
  - **Details:** Pillow is an advanced library for image processing in Python, which extends the capabilities of the original PIL (Python Imaging Library). It supports opening, manipulating, and saving many different image file

formats. In this project, Pillow is used to read and display images within the Tkinter GUI.

- After processing an X-ray image, Pillow helps resize and convert the image to a format that can be displayed in the Tkinter interface, ensuring that users can visually inspect the processed results.

- **cv2 (OpenCV):**
  - **Purpose:** Performs image processing tasks.
  - **Details:** OpenCV (Open Source Computer Vision Library) is a comprehensive library for computer vision and image processing. It offers numerous functions for image manipulation, analysis, and transformation. In this project, OpenCV is used to read X-ray images, preprocess them, and apply various image processing techniques to prepare them for fracture detection. Functions such as resizing, grayscale conversion, and filtering are crucial steps before passing the images to the detection model.
- **cvzone:**
  - **Purpose:** Assists in drawing rectangles and adding text to images.
  - **Details:** Cvzone is a utility library that simplifies common computer vision tasks performed with OpenCV. It provides easy-to-use functions for drawing shapes, adding text, and other visual annotations on images. In this project, Cvzone is used to overlay detection results on X-ray images. For example, when a fracture is detected, cvzone helps draw bounding boxes around the detected areas and label them with the appropriate class and confidence scores, making the output more informative and user-friendly.
- **ultralytics.YOLO:**
  - **Purpose:** Utilized for loading and applying the YOLO object detection model.
  - **Details:** Ultralytics provides a streamlined interface for working with YOLO models, which are state-of-the-art object detection models. The YOLO (You Only Look Once) model can detect objects in images in real-time with high accuracy. In this project, the Ultralytics library is used to load a pre-trained YOLO model specifically trained for detecting hand fractures. It simplifies the process of applying the model to new images, handling the underlying complexity of the detection algorithm.

## 3.2 Model Training

- **Training Process:**
    - **Dataset Preparation:** The YOLO model is trained on a dataset of annotated hand X-ray images. These images are labelled to identify regions containing fractures. This annotation is crucial as it provides the ground truth that the model learns from.
    - **Epochs and Learning:** Training involves running the model through multiple epochs. An epoch is one complete pass through the entire training dataset. During each epoch, the model makes predictions on the training images, calculates errors based on the ground truth, and adjusts its parameters to reduce these errors.
    - **Optimization:** The training process involves fine-tuning the model's parameters, such as learning rate, batch size, and other hyperparameters, to optimize its performance. The goal is to minimize the loss function, which measures how far the model's predictions are from the actual labels.

- **Techniques Used:**

- **Data Augmentation:** To improve the model's robustness and generalizability, data augmentation techniques are applied. These techniques involve transforming the training images in various ways, such as rotating, flipping, scaling, and adding noise. This helps the model learn to recognize fractures in different orientations and conditions, making it more effective in real-world scenarios.
- **Model Validation:** Throughout the training process, the model's performance is validated using a separate set of images that were not used in training. This helps monitor the model's accuracy and adjust the training process to prevent overfitting, where the model performs well on the training data but poorly on new, unseen data.

## 3.3 YOLO Model

The YOLO model is central to the project, loaded from a specified path (`best (3).pt`). This model has been trained to detect hand fractures, distinguishing between various fracture types such as angle fractures, linear fractures, and more complex fracture patterns.

## 3.4 Precautions Set

A dictionary containing precautionary guidelines for different types of hand fractures is used. This includes advice on how to handle various fracture types, ensuring users receive the appropriate care instructions.

## 3.5 Image Processing

When the "Upload and Detect" button is pressed, the process_image function is called. This function:

- Opens a file dialog for the user to select an image.
- Reads the selected image using OpenCV (cv2.imread).
- Processes the image with the YOLO model to detect fractures.

## 3.6 Fracture Detection and Precautions

The YOLO model returns detection results, including bounding boxes, confidence scores, and class labels. For each detected fracture with a confidence score above 0.5, a rectangle is drawn around the detected area, and the corresponding precautionary guidelines are retrieved and displayed. If no fractures are detected, a message "No fractures detected! Keep smiling. 😀 " is shown.

## 3.7 Displaying Results

The processed image, with bounding boxes and labels, is displayed in the GUI. The precautionary guidelines are displayed in a scrolled text box.

# 4. CODE STRUCTURE

The main components of the project are structured as follows**:**

### GUI Initialization

```
import tkinter as tk
from tkinter import filedialog, scrolledtext
from PIL import Image, ImageTk
import cv2
import cvzone
from ultralytics import YOLO
```

```
# Initialize GUI
window = tk.Tk()
window.title("Hand Fracture Detection")
window.geometry("800x600")
```

**Load YOLO Model**:

```
# Load YOLO model
model = YOLO("path/to/best (3).pt")
```

**Precautionary Guidelines**:

```
precautions = {
    'angle': "Precaution for angle fracture...",
    'fracture': "Precaution for fracture...",
    'line': "Precaution for line fracture...",
    'messed_up_angle': "Precaution for messed-up angle fracture..."
}
```

**Image Processing Function**:

```
def process_image():
 # Open file dialog to select image
 file_path = filedialog.askopenfilename()
 if file_path:
 # Read image
 img = cv2.imread(file_path)
 # Detect fractures
 results = model(img)
 # Display results and precautions
 display_results(img, results)
```

**Display Results Function**:

```
def display_results(img, results):
for result in results:
if result['confidence'] > 0.5:
# Draw bounding box and label
cvzone.cornerRect(img, result['bbox'])
cvzone.putTextRect(img,result['label'],(result['bbox'][0],
result['bbox'][1] - 10))
```

```
 # Display precautions
precaution_text.insert(tk.END,precautions.get(result['label'],"No
specific precaution available."))
else:
precaution_text.insert(tk.END,"Nofractures detected! Keep smiling. 😀")

# Convert image to display in Tkinter
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = Image.fromarray(img)
    img_tk = ImageTk.PhotoImage(img)
    img_label.config(image=img_tk)
    img_label.image = img_tk
```

**GUI Elements**:

```
# Upload button
upload_btn = tk.Button(window, text="Upload and Detect",
command=process_image)
upload_btn.pack()

# Image display label
img_label = tk.Label(window)
img_label.pack()

# Precautionary text box
precaution_text = scrolledtext.ScrolledText(window, width=60, height=10)
precaution_text.pack()
```

**Run GUI Loop**:

```
window.mainloop()
```

## 5. RESULTS

The system successfully detects hand fractures and provides relevant precautionary guidelines. The GUI allows users to easily upload images and view the detection results, enhancing the user experience, here is some results of the project.

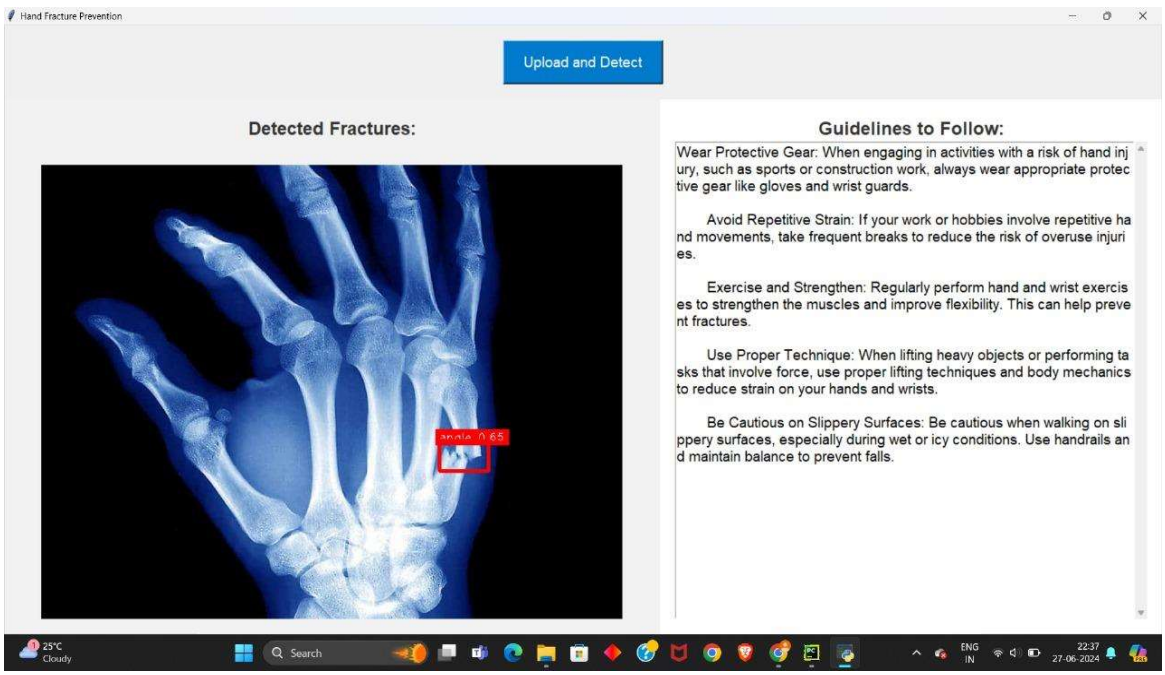


Fig5.1: Utilizing image processing to identify potential fractures in hand X-rays.

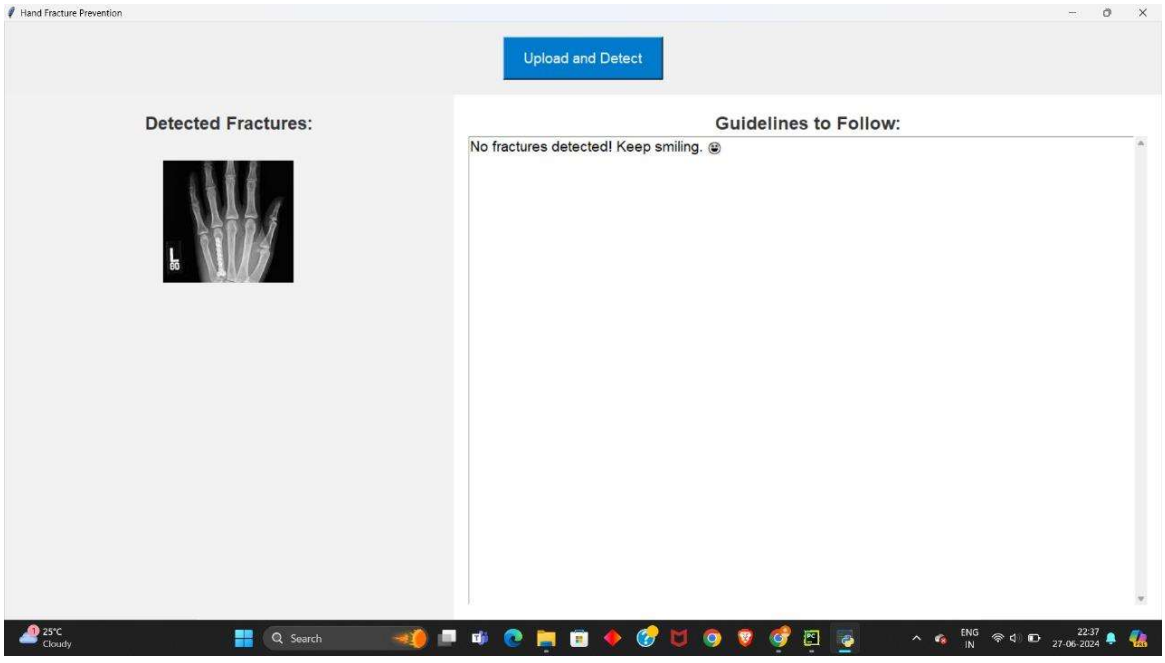


Fig5.2: Hand X-ray analysis using image processing reveals no evidence of fracture.

# 6. CONCLUSION

This project demonstrates the effective application of image processing and machine learning in medical diagnostics. By detecting hand fractures from X-ray images and providing corresponding precautionary guidelines, the system assists in early diagnosis and appropriate care, potentially improving patient outcomes. Future improvements could include expanding the model to detect other types of fractures and integrating more detailed medical advice.