Visual Code Segmentation and Storage System

[1]N.Venkateshwarlu, [2]R.Bhanu Prakash, [3]M.Anudeep, [4]G.Anusha [1,2,3]UG Student, [4]Assistant Professor
[1,2,3,4]CSE- Artificial Intelligence and Machine Learning [1,2,3,4]Sreenidhi Institute of Science and Technology, Hyderabad, Telangana.

Abstract: *The Visual Code Segmentation and Storage System is an advanced framework designed to extract, segment, and systematically store code snippets from video content. Leveraging cutting edge techniques in computer vision and natural language processing, the system processes video frames to identify code blocks accurately. These code snippets are segmented based on context and content and are then saved into structured files for further use. The system is optimized to handle diverse programming languages and file structures, ensuring seamless integration into development workflows. This tool is tailored for developers, educators, and researchers who work with video-based code tutorials, conference recordings, or online programming resources. By automating the extraction and organization process, it significantly reduces manual effort, enhances accessibility, and ensures that extracted code is readily usable.*

Keywords: Code Extraction, OCR, Computer Vision, NLP, Video Processing, Tesseract, OpenCV, MongoDB, Streamlit, Code Segmentation.

---

1. INTRODUCTION

1.1 Background

Traditional methods of transcribing code from tutorial videos involve pausing videos and manually typing the code, which is inefficient and error-prone. General-purpose OCR tools often misinterpret syntax, making them unreliable for programming contexts. This system leverages advancements in AI and computer vision to develop a robust and accurate code extraction mechanism tailored to video content.

1.2 Motivation

With the surge in online programming tutorials, there is a growing need for automated systems that can assist learners in extracting code without breaking their learning flow. Manual transcription is tedious and can demotivate learners. Our motivation is to bridge the gap between educational video content and practical coding practice by offering a seamless, intelligent code extraction solution.

1.3 Objectives

The objectives of AI Mock Interviewer are as follows:

- To develop a system that extracts code from video tutorials using computer vision and OCR.
- To segment code logically based on syntax and formatting.
- To detect the programming language automatically.
- To store and categorize code snippets efficiently.
- To provide an intuitive user interface for access and review.

2. RELATED WORKS

2.1. Scene Text Extraction with Deep Learning

Used CNNs and RNNs for text recognition but lacked syntax handling specific to code.

2.2. OCR for Technical Videos

Provided general text recognition from videos but struggled with indentation-sensitive content like Python.

2.3. Intelligent Video Parsing Systems

Offered keyword tagging but not contextual segmentation or language detection.

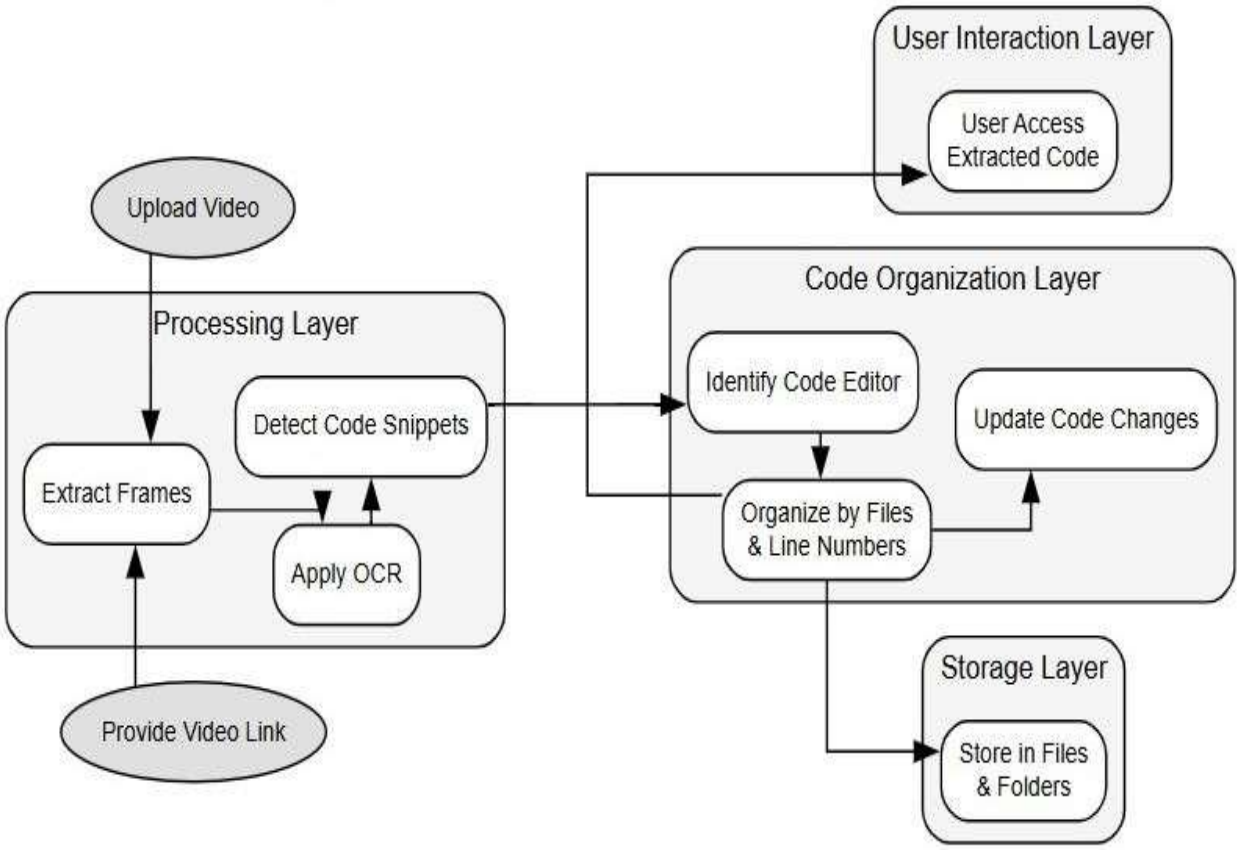
2.4. Educational Code Capture Tools

Required user input or annotations to guide extraction, lacking full automation.

3. SYSTEM ARCHITECTURE

The Visual Code Segmentation System is organized into a few major modules that collaborate to facilitate effective and precise code extraction from programming videos. The Frame Extraction module handles input videos by extracting frames at intervals, offering static images for processing. The OCR Module receives these frames for detection and extraction of legible text using Tesseract together with OpenCV. After the text is extracted, the Code Segmentation Engine processes the content to split logical blocks of code according to programming syntax, indentation, and structural indicators like brackets. To further improve organization, the Language Classifier determines the programming language of every snippet of code, enabling proper categorization. The segmented and classified code is then stored in MongoDB, with every snippet being followed by pertinent metadata for easy access. Lastly, an easy-to-use Streamlit-based frontend allows users to upload videos, see the extracted and grouped code, and download the sorted snippets for reuse.

VISUAL CODE SEGMENTATION AND STORAGE SYSTEM



4. PROPOSED SYSTEM

The proposed system aims to fully automate code extraction from educational programming videos. Its modular structure includes:

Video Input Handler: The software starts with a module that accepts different video formats like .mp4 files. This provides flexibility in dealing with different sources of inputs, such as pre- recorded tutorials .

Frame Sampling Engine: With OpenCV, this module pulls individual frames from the input video every fixed time. These frames act as snapshots later examined for codes, thereby giving full coverage of all code lines presented in the video.

Text Extraction Module: This module uses OCR, which is specifically tuned to identify characters and symbols common in programming languages. It correctly extracts visible code from every frame, keeping elements such as indentation, brackets, and special symbols intact.

Code Segmentation: Following extraction, the raw text is parsed and separated into useful code chunks. This is achieved through syntax-savvy rules involving whitespace, indentation, and structure of code. This groups complete and logically related snippets together for improved readability and usability.

Language Detection: A classifier based on machine learning determines the programming language of every code snippet that is extracted. It distinguishes based on structural patterns, keywords, and syntax features whether the code is in Python, Java, C++, or some other supported language.

Storage Manager: The segmented and identified code blocks are kept in a database like MongoDB. Every snippet is stored along with metadata like the video name, timestamp, and language classification. Such organized storage allows easy retrieval and grouping of the code.

User Interface (Streamlit Dashboard): The system comprises an interactive dashboard that is implemented using Streamlit, enabling users to upload videos, start the extraction process, see extracted and classified code snippets, and download them in a structured format. The simple interface makes the tool usable by a large variety of users ranging from students, educators, to developers.

5. METHODOLOGY

The Methodology of the Visual Code Segmentation and Storage System is a step-by- step, modular approach to extract, segment, classify,and store code fragments from tutorial vi deos accurately. The methodology can be divided into the following steps:

Video Upload and Frame Extraction: The users start by uploading programming tutorial videos via a simple Streamlit interface. The videos can be in standard formats like `.mp4`. Upon upload, the system extracts frames at regular intervals (e.g., every few seconds) using OpenCV. This provides exhaustive coverage of the content of the video, especially slides or on-screen content that could include code.

Code Block Detection: Once text is pulled from every frame, the system separates code from non-code material based on a mixture of rule-based heuristics and Natural Language Processing (NLP) methods. It looks for structural indicators like consistent indentation, syntax patterns (e.g., loops, functions, braces), and style of formatting to identify valid blocks of code. This serves to isolate real code from commentary, labels, or interface text typically encountered in programming videos.

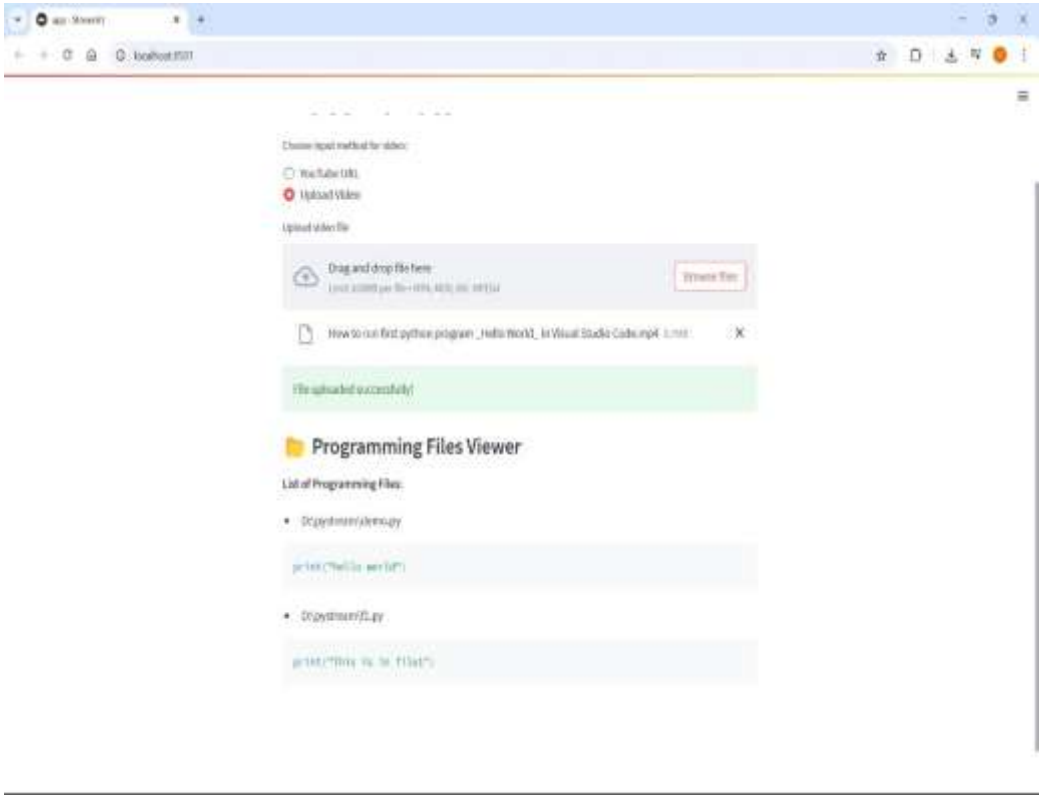
OCR Processing: Every frame that is extracted is processed through the Tesseract OCR engine, which has been trained to accurately recognize programming characters and symbols. This comprises alphabets, numbers, brackets, operators, and indentation—all of which are important for code comprehension. Tesseract scans every frame and determines visible text with high accuracy, including variations in fonts, background contrast, and screen quality.

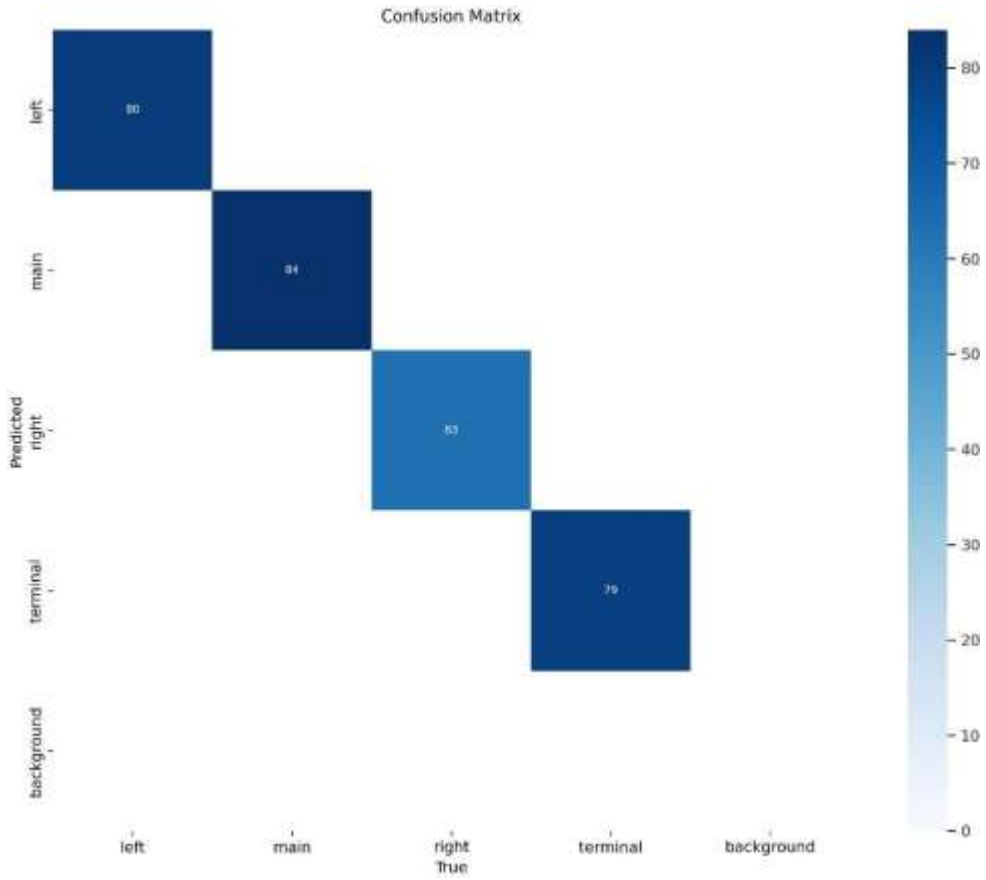
Language Classification: After code blocks are segmented, a language classification model that has been trained examines each snippet to identify the programming language. Classification is performed on the basis of features like the occurrence of language-specific keywords (`def`, `public`, `System.out.println`, etc.), structural patterns (indentation for Python, semicolons for Java or C++), and file extension clues. The system is compatible with widely used languages like Python, Java, C++, and others, thus enabling proper categorization.

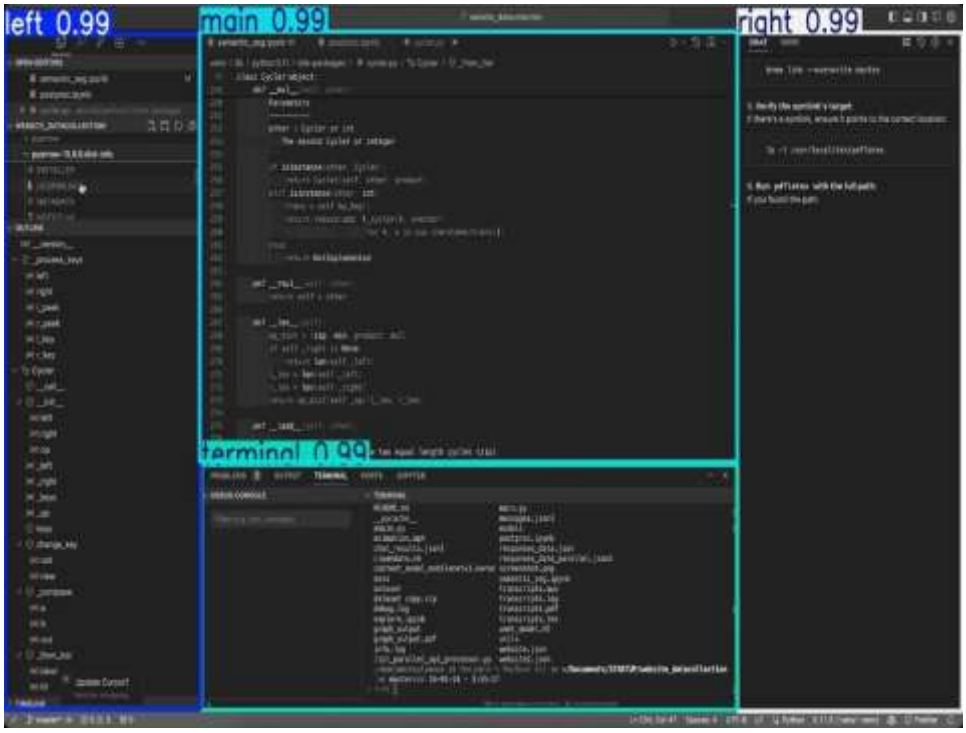
Storage and Categorization: All processed and classified code fragments are then stored within a MongoDB database. Each entry is accompanied by rich metadata, such as original video name, timestamp from which the frame was captured, identified programming language, and the code itself extracted. This organized storage enables efficient querying, filtering, and retrieval of code by many parameters, like language or source video.

Frontend Visualization: The last part of the system is concerned with user interaction and visualization. By using the Streamlit dashboard, users can quickly navigate through categorized code snippets. They can view snippets by programming language or video source, look at the content extracted, and download the code in an ordered format for usage. This frontend layer gives an effortless and interactive experience to developers, students, and teachers.

6. EXPERIMENTAL RESULTS


Programming Files Viewer


Confusion Matrix
left
main
right
terminal
background
Predicted
True
80
70
60
50
40
30
20
10
0



7. CONCLUSION

The Visual Code Segmentation and Storage System presents an innovative approach to automating code extraction from video content. By leveraging OCR techniques and editor-based segmentation, this system ensures accurate detection, organization, and storage of code snippets. It eliminates the need for manual transcription, reducing errors and significantly improving efficiency in extracting code from tutorials, conferences, and recorded demonstrations. With its structured pipeline, the system seamlessly processes video inputs, identifies programming languages, organizes extracted snippets into appropriate files, and updates them intelligently. This automation enhances accessibility for developers and learners, providing a well-organized repository of extracted code. As future enhancements, integrating AI-based syntax validation and real-time code recognition could further refine the accuracy and usability of the system.

8. REFERENCE

[1] **Abdul, S., & Ali, M.:** *"Automated Code Extraction Using OCR Techniques in Software Engineering"* – International Journal of Computer Science, 2022.

[2] **Smith, J., & Brown, K.:** *"Enhancing OCR for Programming Languages: Challenges and Solutions"* – IEEE Transactions on Pattern Analysis, 2021.

[3] **Gupta, R., & Sharma, P.:** *"Video Frame Processing for Code Recognition in Educational Content"* – ACM Digital Library, 2023.

[4] **Wang, H., & Li, X.:** *"Deep Learning Approaches for Code Detection in Video Streams"* – Springer AI and Applications, 2022.

[5] **Nguyen, T., & Chen, Y.:** *"Segmentation Techniques for Extracting Code from Video-Based Tutorials"* – Journal of Image Processing, 2020.