|  |  |  |
| --- | --- | --- |
| **S. No** | **TITLE** | **PAGE No** |
| **1** | **INTRODUCTION** |  |
|  | 1.1 ORGANIZATION PROFILE |  |
|  | 1.2 EXISTING SYSTEM |  |
|  | 1.3 PROPOSED SYSTEM |  |
|  | 1.4 SCOPE |  |
|  | 1.5 HARDWARE REQUIREMENTS |  |
|  | 1.6 SOFTWARE REQUIREMENTS |  |
| **2** | **PROBLEM STATEMENT** |  |
|  | 2.1 REAL-WORLD CHALLENGES FOR VISUALLY IMPAIRED INDIVIDUALS |  |
|  | 2.2 LIMITATIONS OF EXISTING SOLUTIONS |  |
|  | 2.3 SPECIFIC ISSUES THIS PROJECT AIMS TO SOLVE |  |
| **3** | **OBJECTIVES** |  |
|  | 3.1 MAIN OBJECTIVES |  |
|  | 3.2 KEY FUNCTIONAL IMPLEMENTATIONS |  |
|  | 3.2.1 REAL-TIME DISTANCE ESTIMATION |  |
|  | 3.2.2 NAVIGATION DECISION LOGIC |  |
|  | 3.2.3 IMAGE HASHING FOR FRAME SIMILARITY FILTERING |  |
|  | 3.2.4 OFFLINE OPERATION |  |
| **4** | **SYSTEM ARCHITECTURE** |  |
|  | 4.1 OVERVIEW |  |
|  | 4.2 KEY COMPONENTS |  |
|  | 4.2.1 CAMERA INPUT |  |
|  | 4.2.2 OBJECT DETECTION (YOLO12n) |  |
|  | 4.2.3 IMAGE HASHING |  |
|  | 4.2.4 DISTANCE ESTIMATION |  |
|  | 4.2.5 ZONAL FRAME DIVISION |  |
|  | 4.2.6 NAVIGATION DECISION LOGIC |  |
|  | 4.2.7 SMALL LANGUAGE MODEL (SLM) |  |
|  | 4.2.8 VOICE OUTPUT |  |
|  | 4.3 FLOWCHART |  |
|  | 4.4 DATA FLOW |  |
|  | 4.5 ADVANTAGES OF THIS ARCHITECTURE |  |
| **5** | **TOOLS AND TECHNOLOGIES USED** |  |
|  | 5.1 YOLO12N – OBJECT DETECTION ENGINE |  |
|  | 5.2 SMOLLM2 1.7B – SMALL LANGUAGE MODEL |  |
|  | 5.3 PYTHON – PROGRAMMING LANGUAGE |  |
|  | 5.4 OPENCV – COMPUTER VISION TOOLKIT |  |
|  | 5.5 PYTORCH – DEEP LEARNING FRAMEWORK |  |
| **6** | **IMPLEMENTATION** |  |
|  | 6.1 DATA ACQUISITION & SIMULATION SETUP |  |
|  | 6.2 YOLO12n MODEL INTEGRATION |  |
|  | 6.3 SMALL LANGUAGE MODEL (SMOLLM2) USAGE |  |
|  | 6.4 CODE FLOW AND PROCESSING PIPELINE |  |
|  | 6.5 INPUT TO OUTPUT PIPELINE |  |
|  | 6.6 CHALLENGES FACED |  |
| **7** | **RESULTS** |  |
|  | 7.1 VISUAL OUTPUT – OBJECT DETECTION & POSITIONING |  |
|  | 7.2 NATURAL LANGUAGE GUIDANCE VIA SLM |  |
|  | 7.3 AUDIO OPTIMIZATION AND FEEDBACK CONTROL |  |
|  | 7.4 PERFORMANCE AND RESPONSIVENESS |  |
|  | 7.5 SYSTEM EVALUATION & OBSERVATIONS |  |
|  | 7.6 USER-CENTERED IMPACT |  |
| **8** | **CONCLUSION** |  |
|  | 8.1 RECAP OF THE WORK |  |
|  | 8.2 KEY OUTCOMES |  |
|  | 8.3 BENEFITS OF THIS APPROACH |  |
| **9** | **FUTURE ENHANCEMENTS** |  |
| **10** | **REFERENCES** |  |
| **11** | **APPENDIX** |  |
|  | 11.1 SAMPLE CODE SNIPPETS |  |
|  | 11.2 OUTPUT PICTURES |  |

# ABSTRACT

Navigating the physical world independently poses significant challenges for visually impaired individuals, particularly in unfamiliar or dynamic environments. Everyday tasks such as walking through crowded spaces or identifying obstacles can become difficult without constant external assistance.

Existing assistive technologies often rely on expensive hardware or fixed rule-based systems. While they offer some level of support, they frequently lack the flexibility and adaptability required for real-time, context-aware decision-making.

This project presents a simulation of an AI-powered blind navigation system. It combines advanced object detection with contextual, natural-language guidance using a Small Language Model (SLM). This integration aims to create a more responsive and intelligent support system for the visually impaired.

The system employs the efficient YOLO12n model to detect and identify objects in real time. These visual insights are then processed by a lightweight language model, SmolLM2, which converts them into clear and meaningful navigation instructions.

The entire setup is simulated to reflect how a visually impaired person might receive guidance in structured environments, such as corridors or indoor settings. The use of AI for both perception and instruction delivery ensures a smoother and more natural user experience.

By merging object recognition and language generation on a compact, edge-compatible system, the project demonstrates a viable approach for smart assistive tools. It highlights the potential to develop low-cost, high-impact mobility aids that improve accessibility and independence for visually impaired individuals.

# **CHAPTER 1: INTRODUCTION**

# **1.1 Organization Profile**

# Data Aces is a cutting-edge technology solutions provider focused on delivering advanced AI, data science, and automation services to a wide range of industries. With a strong emphasis on innovation, Data Aces empowers organizations to harness the full potential of their data through intelligent systems, machine learning, and real-time analytics.

# The company provides end-to-end AI solutions including model development, deployment, and integration with edge devices. With a mission to bridge the gap between theoretical AI models and practical real-world applications, Data Aces has made notable contributions in fields such as smart automation, accessibility tech, and embedded AI.

# Driven by a team of domain experts and engineers, the company focuses on scalable, sustainable, and user-centric designs. Their work in inclusive technologies highlights a commitment to solving real-world problems, such as navigation for visually impaired individuals—an area where AI can truly improve lives.

# **1.2 Existing System**

# Visually impaired individuals often rely on traditional mobility aids such as white canes or guide dogs for assistance. While these tools provide basic navigation support, they have significant limitations:

# They cannot detect objects beyond their physical reach.

# They lack dynamic context-awareness.

# They do not offer directional or situational feedback.

# In recent years, a few AI-based assistive technologies have emerged, but many of them are either too expensive, depend heavily on cloud infrastructure, or require specialized hardware like LiDAR or high-performance GPUs. These limitations restrict their usability, especially in portable or real-time edge scenarios. Most systems provide detection data but do not generate actionable instructions in human-readable language.

# **1.3 Proposed System**

# This project proposes an AI-powered blind navigation simulation that integrates real-time object detection with human-like language guidance. The goal is to simulate how a visually impaired individual can navigate a structured environment with intelligent assistance.

# The system employs YOLO12n (nano version) for lightweight, fast, and accurate object detection. For natural instruction generation, it uses SmolLM2, a small language model capable of producing human-understandable, context-aware navigation prompts suitable for low-power environments.

# Key Features:

# Real-time object detection using camera input.

# Context-aware guidance using a lightweight language model.

# Simulation-based testing in a controlled environment.

# Designed for deployment on edge devices (e.g., Raspberry Pi, mobile systems).

# Focus on accessibility, affordability, and future real-world scalability.

# **1.4 Scope**

# The scope of this project includes:

# Designing a simulated environment for visually impaired navigation.

# Integrating YOLO12n for object detection in real-time video feeds.

# Implementing SmolLM2 to convert visual data into natural language instructions.

# Ensuring the system performs efficiently on edge-compatible hardware.

# Demonstrating how AI can enhance accessibility in indoor environments such as hallways, classrooms, or public buildings.

# Potential future enhancements include:

# Real-world testing with cameras and microphones.

# Integration of voice output for auditory feedback.

# Expansion to outdoor environments using GPS and additional sensors.

# **1.5 Hardware Requirements**

# To effectively simulate and test the system, the following hardware configuration is recommended:

# Processor: Intel Core i5 / ARM Cortex-A72 (or equivalent)

# RAM: 8 GB or higher

# Storage: 256 GB SSD (for model and data storage)

# Camera: USB or Pi Camera for video input

# Display: Monitor for debugging/visual output (optional)

# Input: Keyboard and mouse (for control and testing)

# **1.6 Software Requirements**

# The proposed system depends on the following software components:

# Operating System: Windows 10 / Linux (Ubuntu recommended for edge)

# Programming Language: Python 3.10+

# Libraries:

# Ultralytics YOLO12n

# OpenCV (for image processing)

# Torch (for deep learning)

# Transformers / custom tokenizer (for SmolLM2)

# IDE: VS Code / Jupyter Notebook/ Google Colab

**2. Problem Statement**

**2.1 Real-World Challenges for Visually Impaired Individuals**

Visually impaired people often face numerous challenges when navigating independently, especially in complex or unfamiliar environments. Common tasks like walking down a hallway, crossing a street, or entering a building can become hazardous without adequate support. These challenges include:

* Difficulty detecting and avoiding dynamic obstacles (e.g., people, furniture, vehicles).
* Inability to recognize spatial layouts or changing environments.
* Dependency on others for safe mobility and orientation.
* Limited situational awareness in noisy or crowded places.
* Lack of real-time feedback during movement.
  1. **Limitations of Existing Solutions**

Several assistive technologies exist, ranging from traditional tools like white canes to modern AI-based devices. However, these solutions are often inadequate due to various limitations:

* **White canes**: Only detect objects through touch and offer limited range.
* **Guide dogs**: Expensive to train and maintain, with limited availability.
* **GPS-based apps**: Work mostly outdoors and lack object-level awareness.
* **Camera-based systems**: Often rely on large AI models that require cloud connectivity or powerful hardware, making them unsuitable for edge deployment.
* **Rule-based software**: Lacks flexibility in dynamic environments and cannot adapt to unseen obstacles or layouts.

**2.3 Specific Issues This Project Aims to Solve**

This project targets the development of a **simulated AI-based blind navigation system** that overcomes many of the issues listed above by combining **object detection** and **natural language guidance**. The specific challenges addressed are:

* **Real-Time Guidance**: Providing timely and actionable instructions to the user as the environment changes.
* **Efficient Object Detection**: Using YOLO12n to identify obstacles and key elements in the environment with low latency.
* **On-Device Processing**: Ensuring the system runs on resource-constrained devices by utilizing lightweight models like SmolLM2.
* **Contextual Language Output**: Generating human-like, easy-to-understand navigation instructions instead of raw object data.
* **Simulation of Realistic Scenarios**: Building a virtual setup that mirrors real-life navigation situations for testing and demonstration.

This project aims to bridge the gap between vision-based perception and language-based communication, simulating how an intelligent assistant could help visually impaired individuals navigate safely and independently—without relying on expensive or bulky hardware.

**CHAPTER 3: OBJECTIVES**

**3.1 Main Objectives**

The primary goal of this project is to simulate an AI-powered navigation system for visually impaired individuals that works in real time and does not rely on internet connectivity. The system combines computer vision and natural language processing to provide intelligent, spoken guidance based on environmental context.

**Specific Objectives:**

* To **simulate a blind navigation system** using AI techniques in a structured environment.
* To **implement YOLO12n** (nano version) for real-time object detection with low computational overhead.
* To **incorporate a small language model (SmolLM2)** for generating human-readable, contextual navigation instructions which has 1.7B parameters.
* To **visualize and simulate the system’s guidance decisions** by processing an uploaded video file, where real-time object detection and navigation logic are applied frame-by-frame.
* To **use image hashing** to filter out redundant frames, improving speed and reducing processing load.
* To **run the entire system offline**, enabling use in areas without internet access.
* To **estimate object distances** from the camera using focal length and known object widths.
* To **analyze object positions and distances** and recommend safe movement directions using a scoring-based decision algorithm.

**3.2 Key Functional Implementations**

**3.2.1 Real-Time Distance Estimation**

The system uses the pinhole camera model to estimate how far an object is from the camera. Each object’s actual width (in meters) is compared to its width in pixels in the captured image using the formula:

A predefined dictionary of real-world object widths is used (REAL\_WIDTHS), and the focal length (FOCAL\_LENGTH = 700) was experimentally calibrated. This allows the system to make proximity-based decisions even on simple camera hardware.

**3.2.2 Navigation Decision Logic**

Using the detected objects, their **positions** (left, center, right), and their **estimated distances**, the system computes a weighted risk score. More dangerous objects (like buses or people) contribute more to the threat level than smaller or stationary ones (like backpacks or cups).

**Scoring Logic:**

* The closer and more dangerous an object, the higher its weight.
* Each position (left, center, right) accumulates a score.
* The system decides to **move forward**, **stop**, or **adjust direction** based on score differences.

**3.2.3 Image Hashing for Frame Similarity Filtering**

To avoid redundant computations and reduce lag, the system implements an **image hashing technique** (like average hash or perceptual hash). This helps identify whether the current frame is too similar to the previous one. If so, it skips processing and saves resources, allowing smoother performance on low-power devices.

**3.2.4 Offline Operation**

A key design feature of the system is that **it works entirely offline**, making it viable in locations without internet connectivity. All models (YOLO12n and SmolLM2) are loaded locally, and no cloud services are required for detection or instruction generation.

**CHAPTER 4: SYSTEM ARCHITECTURE**

**4.1 Overview**

At the heart of the system is a real-time video processing pipeline that simulates a visually impaired person's point of view. The system captures video frames, detects nearby obstacles, estimates their distances and positions, and generates actionable guidance using a compact language model. These instructions are then relayed to the user through speech output at regular intervals. This ensures continuous situational awareness and helps users confidently navigate complex environments such as corridors, sidewalks, and public spaces.

The modular design also makes it highly adaptable, allowing each core component — such as object detection, decision logic, and language generation — to be improved independently as needed.

**4.2 Key Components**

**4.2.1 Camera Input**

* A webcam or mobile camera serves as the user’s “eyes,” capturing real-time video of the surrounding environment.
* For simulation purposes, recorded videos can also be used during testing and debugging.

**4.2.2 Object Detection (YOLO12n)**

* The YOLO12n (nano) model is a lightweight deep learning model optimized for real-time object detection on resource-constrained devices.
* It identifies multiple classes of objects (like people, vehicles, obstacles, etc.) and provides their bounding box coordinates.
* Each frame is processed to extract these visual cues, forming the basis for environmental understanding.

**4.2.3 Image Hashing**

* To reduce computational load, the system uses perceptual hashing (like average hash or difference hash) to skip frames that are visually similar.
* This ensures that only meaningfully different frames are analyzed, speeding up processing while maintaining accuracy.

**4.2.4 Distance Estimation**

* The system estimates how far each object is from the user using the formula:
* *D = (W x F) / P*

D = Distance

W = Real Width

F = Focal Length

P = Detected Width in Pixels

* Known real-world object dimensions (like average human width) and the camera's focal length are used for calibration.
* This allows the system to prioritize closer obstacles and calculate risk zones.

**4.2.5 Zonal Frame Division**

* Each frame is horizontally divided into three zones: **Left**, **Center**, and **Right**.
* Based on where the object's bounding box falls, its relative position is categorized.
* This zonal division allows for intelligent navigation suggestions (e.g., “turn slightly left” if an obstacle is in the center).

**4.2.6 Navigation Decision Logic**

* Using detected object types, positions, and estimated distances, the system scores the threat level in each zone.
* Based on this analysis, the system decides whether the user should:
  + Continue forward
  + Slightly turn left/right
  + Sharply turn
  + Stop completely
* This logic mimics human decision-making in unfamiliar or obstructed environments.

**4.2.7 Small Language Model (SLM)**

* A small, local language model (e.g., SmolLM2) generates natural-language guidance.
* Instead of simple directional outputs, it produces user-friendly phrases like:
  + “Move slightly left, there’s a clear path.”
  + “Obstacle ahead in the center. Consider turning right.”
* The SLM ensures the guidance is intuitive and easier to trust.

**4.2.8 Voice Output**

* The final navigation instruction is spoken aloud using a text-to-speech (TTS) engine.
* Speech is triggered every few frames to avoid redundancy, simulating natural real-time feedback for the user.

**4.3 Flowchart**

**Camera Input**

**Video Frame Extraction & Hash Filtering**

**YOLO12nObject Detection**

**Distance Estimation + Zonal Positioning (Left / Center / Right)**

**Threat Analysis & Navigation Logic**

**Small Language Model (Instruction Generation)**

**Text-to-Speech Output**

This modular pipeline ensures that each step contributes meaningfully to the final instruction, with minimal delay or computational burden.

**4.4 Data Flow**

Here's a detailed breakdown of how data moves through the system:

1. **Frame Input**:
   * Captures or reads video input.
   * Skips redundant frames using perceptual hashing for efficiency.
2. **Detection & Localization**:
   * YOLO12n detects objects, tags them with labels, and draws bounding boxes.
3. **Distance Computation**:
   * For each object, the distance is estimated using bounding box width and a calibrated formula.
4. **Zone Threat Analysis**:
   * Each object is assigned to a zone (left, center, right).
   * Distances are used to rank threats in each zone.
5. **Guidance Decision**:
   * The system uses a rule-based logic to decide whether to move forward, stop, or turn.
   * Closest zones with the least threat are prioritized.
6. **SLM Instruction**:
   * A small language model turns the numeric and positional data into human-friendly text.
7. **Voice Feedback**:
   * The final guidance is spoken aloud every few seconds to ensure user clarity without overwhelming them.

**4.5 Advantages of This Architecture**

* **Offline Operation**: Entirely functional without internet — suitable for real-world assistive use.
* **Real-Time Feedback**: Minimally delayed response, giving the user timely and relevant guidance.
* **Lightweight & Efficient**: Optimized for edge devices using YOLO12n and frame filtering.
* **Human-Like Communication**: Leverages language models to produce natural and comfortable guidance.
* **Modular Design**: Easily upgradable components — detection, reasoning, and language modules can evolve independently.

**CHAPTER 5: Tools and Technologies Used**

**5.1 YOLO12n – Object Detection Engine**

YOLO12n (You Only Look Once, nano version) functions as the backbone for object detection within the system. It is a lightweight neural network model specifically optimized for real-time performance on devices with limited computational power.

* **High-Speed Inference**: With its streamlined architecture, YOLO12n enables rapid object detection with minimal latency, making it suitable for real-time applications.
* **Edge-Friendly Design**: The nano variant of YOLO12n is engineered for use on embedded systems, where memory and processing capacity are limited.
* **Accurate Detection**: Despite being lightweight, YOLO12n maintains a high level of detection accuracy for a variety of everyday objects, which is crucial for safety in navigation scenarios.
* **Flexibility**: Easily integrates with OpenCV and PyTorch pipelines, allowing seamless incorporation into the full guidance system.

**5.2 SmolLM2 1.7B – Small Language Model**

SmolLM2 is a compact transformer-based language model comprising 1.7 billion parameters, sourced from Hugging Face. It plays a pivotal role in converting raw object detection data into meaningful, human-readable navigation instructions.

* **Natural Language Generation**: SmolLM2 transforms spatial awareness data into clear verbal instructions, ensuring intuitive guidance for visually impaired users.
* **Offline Capability**: The model can be executed entirely offline, removing reliance on cloud-based APIs or internet connectivity.
* **Optimized for Edge AI**: SmolLM2 balances performance and size, making it ideal for running on devices with constrained resources while still producing high-quality language output.
* **Customizable Behavior**: It can be fine-tuned or prompted with domain-specific templates to generate consistent and context-aware responses.

**5.3 Python – Programming Language**

Python serves as the primary programming language for system development, enabling seamless integration of machine learning models, video processing modules, and voice synthesis capabilities.

* **Developer Productivity**: Python’s clean syntax and readability support fast development cycles and ease of debugging.
* **Rich Ecosystem**: Access to libraries like OpenCV, PyTorch, NumPy, and text-to-speech tools accelerates development and enhances modularity.
* **Cross-Platform**: Python code is highly portable and can be adapted to run on various operating systems and devices.

**5.4 OpenCV – Computer Vision Toolkit**

OpenCV (Open Source Computer Vision Library) is a foundational component used for handling visual data and preprocessing frames before and after object detection.

* **Frame Management**: Handles operations such as video capture, frame resizing, and overlaying bounding boxes with labels and distances.
* **Image Hashing**: Implements perceptual hashing to filter visually similar frames, improving performance by reducing redundant computations.
* **Visualization Support**: Enables real-time visualization of processed frames, which is useful for debugging and demonstrations.
* **Hardware Interface**: Easily connects with built-in or external cameras for live data acquisition.

**5.5 PyTorch – Deep Learning Framework**

PyTorch is the underlying framework used for deploying and running both the object detection and language models in this project.

* **Model Loading and Inference**: Facilitates efficient loading of pre-trained models like YOLO12n and SmolLM2, and executes them with GPU acceleration when available.
* **Dynamic Graph Execution**: PyTorch’s dynamic computation graph allows for more flexible model execution and debugging during runtime.
* **Research to Production**: Provides an ideal environment for both experimental prototyping and robust production deployment.
* **Community and Support**: Widely adopted by the AI community, ensuring strong documentation, tutorials, and community-driven innovation.

This toolset collectively forms a cohesive and powerful pipeline capable of real-time decision-making and assistive interaction. Each tool contributes a critical function from capturing and analyzing visual data to translating complex spatial information into simple verbal instructions making the system an effective and scalable solution for blind or visually impaired navigation.

**CHAPTER 6: IMPLEMENTATION**

**6.1 Data Acquisition & Simulation Setup**

For simulation and testing purposes, video footage representing a user's point of view was used as input. These videos were either captured from a handheld or wearable camera, or sourced from publicly available datasets that mimic real-world walking scenarios. The videos serve as the test environment where object detection and navigation logic could be evaluated safely and repeatedly.

* **Video Format**: MP4, resized to 640x480 for processing efficiency.
* **Frame Skipping**: To optimize performance, the system processes every Nth frame (e.g., every 30th), skipping similar ones using visual hashing.
* **Audio Skipping**: To optimize performance and to avoid the overlapping in every Nth frame (e.g., every 90th).
* **Image Hashing**: Implemented perceptual hashing to avoid redundant processing by filtering out visually similar frames using hash distance thresholds.

**6.2 YOLO12n Model Integration**

The YOLO12n (nano version) object detection model was integrated using the Ultralytics implementation.

* **Detection Output**: Bounding boxes, class labels, and confidence scores.
* **Real-Time Readiness**: The model was optimized to run on CPU or GPU, ensuring minimal latency even on lower-end systems.
* **Post-Processing**: Detected object coordinates were extracted to estimate their spatial positions and distances.

**Position Estimation**:

* The frame is divided into three horizontal zones — Left, Center, and Right — based on object center\_x coordinates.
* Real-world distance is approximated using the formula:

*D = (W x F) / P*

D = Distance

W = Real Width

F = Focal Length

P = Detected Width in Pixels

**6.3 Small Language Model (SmolLM2) Usage**

SmolLM2 (1.7B parameters), hosted on Hugging Face, was employed to transform spatial data into natural language instructions. It enhances the raw output of navigation logic into human-understandable guidance.

* **Prompt-Driven**: The model receives structured prompts containing detected objects, positions, and distances.
* **Instruction Enhancement**: Converts "safe path: center" into conversational directions like *"Move forward, path is clear."*
* **Offline Capability**: Fully executable offline after initial setup, ideal for edge deployment.

**6.4 Code Flow and Processing Pipeline**

**Code Snippets:**

Figure 1:

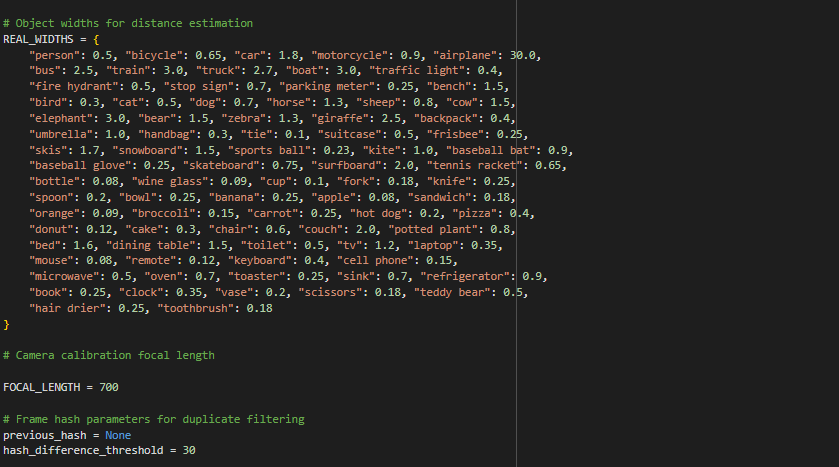
****

Figure 2**:**

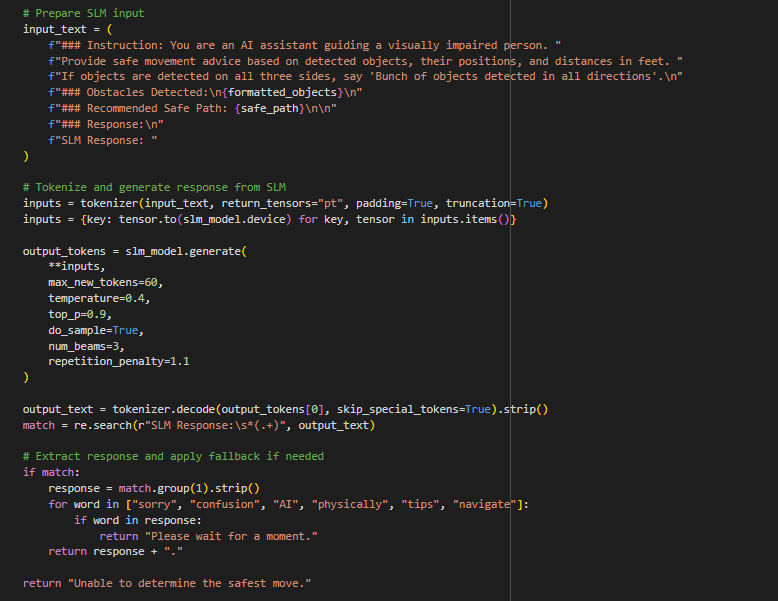
****

Figure 3:

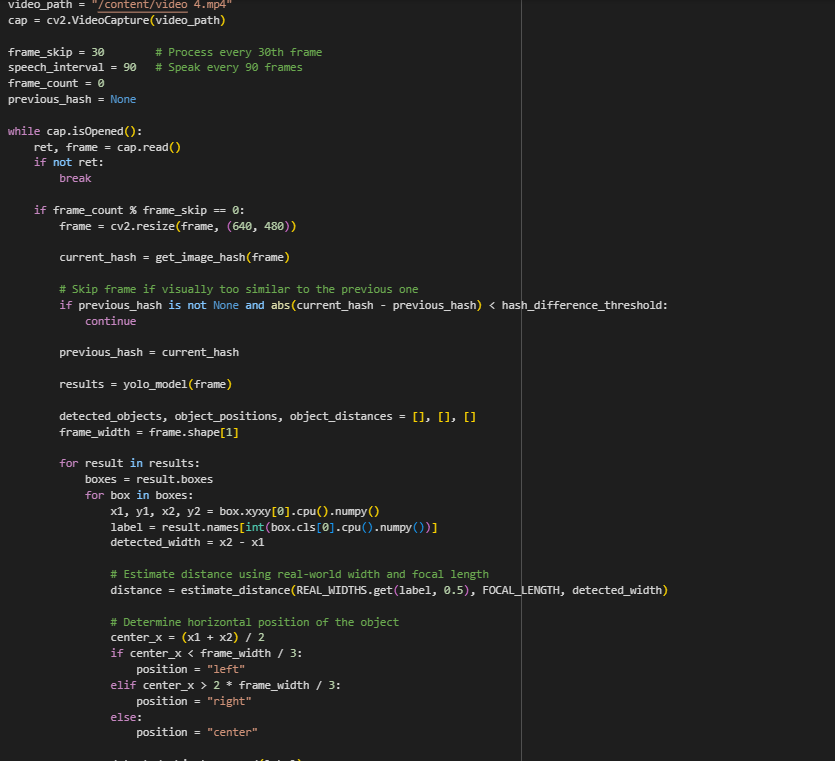
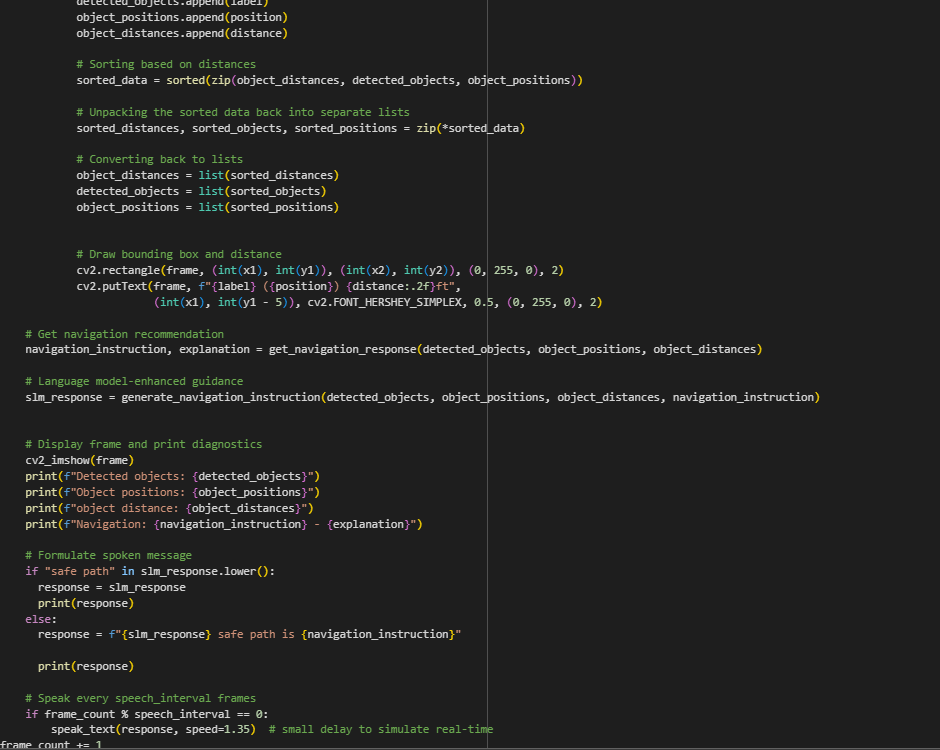
****

Figure 4:

****

The processing pipeline follows a sequential flow:

1. **Video Frame Capture**: A video file is read frame-by-frame using OpenCV.
2. **Image Hash Check**: Only unique-looking frames are processed.
3. **Object Detection**: YOLO12n identifies and classifies objects.
4. **Distance Estimation & Positioning**: Objects are categorized by proximity and location (left, center, right).
5. **Navigation Logic**: A rule-based heuristic ranks the safest direction.
6. **SLM Instruction Generation**: SmolLM2 translates decisions into natural language.
7. **Voice Output**: Instructions are converted to speech using a text-to-speech engine.

**6.5 Input to Output Pipeline**

**Example Walkthrough**

Let’s walk through a sample scenario:

**1. Input Frame:**Frame contains two objects:

* "person" at the center of the frame
* "bike" on the left

**2. Distance Calculation:**

* Person is approximately 3.2 ft away
* Bike is 2.1 ft away

**3. Zone Analysis:**

* Center zone has a moderately close person
* Left zone has a bike, but it’s closer
* Right zone is clear

**4. Navigation Logic Decision:**

* Left: Too close, blocked
* Center: Somewhat blocked
* Right: Free → Preferred path

**5. SLM Output:***“Slightly move to the right. Path ahead is partially blocked by a person.”*

**6.6 Challenges Faced**

During the development of the AI-powered blind navigation system, several technical and design challenges were encountered. Each challenge required careful analysis, experimentation, and problem-solving to ensure the system's reliability, performance, and usability in real-world conditions.

* **Visual Redundancy**

**Problem**: Consecutive frames from the video feed often appeared visually similar, leading to repeated object detection processing and unnecessary computational overhead.  
**Solution:**Implemented **perceptual image hashing** to compare frame similarity and filter out redundant frames. This optimization significantly improved system efficiency and responsiveness.

* **Ambiguous Navigation Scenarios**

**Problem:** In environments with multiple nearby obstacles positioned at roughly equal distances across different zones, the system struggled to make confident navigation decisions.  
**Solution:** Introduced **intermediate directional cues**, such as "Slightly Left" and "Slightly Right", to offer more nuanced and safer guidance when full turns were unnecessary or potentially unsafe.

* **Audio Overlap in Guidance**

**Problem**: Initially, the voice instructions overlapped when multiple prompts were triggered in rapid succession, causing confusion and reducing the clarity of navigation.

**Solution**: Introduced a controlled speech interval by allowing the system to speak once every 90 frames. This throttling mechanism ensures that voice instructions are delivered at regular and manageable intervals, preventing overlap and enhancing the user's ability to follow the guidance in real time.

* **Model Compatibility and Integration**

**Problem:** Integrating YOLO12n (vision model) with SmolLM2 (language model) presented issues due to differences in expected input/output formats and data representations.  
**Solution:** Developed a **well-structured prompt design and data formatting pipeline** to ensure seamless interaction between object detection outputs and natural language generation inputs.

* **Inaccurate Distance Estimation (Monocular Method)**

**Problem**: Initially attempted to estimate object distances using **monocular depth estimation techniques**, which proved unreliable and inconsistent in varied lighting.   
**Solution:** Conducted research and switched to a **focal-length-based geometric formula**, utilizing known real-world object widths and detected bounding box widths to compute distance more accurately. This improved consistency across different frames and object types.

* **Model Selection Difficulty**

**Problem**: Selecting the most appropriate small language model for offline usage posed a challenge due to the need for compactness, speed, and fluency in guidance generation.  
**Solution:** After evaluating multiple lightweight language models, **SmolLM2-1.7B** from Hugging Face was selected based on its balance of performance, size, and compatibility with the target system. This decision was finalized after extensive benchmarking and testing across sample navigation scenarios.

**CHAPTER 7: Results**

**7.1 Visual Output – Object Detection & Positioning**

Upon processing each selected frame, the system overlays bounding boxes on detected objects and annotates them with their class label, position (left/center/right), and estimated distance from the user. This is critical in generating awareness of immediate surroundings.

In one of the test frames, the following detections were made:

* **Objects:** ["person (center)", "bike (left)"]
* **Distances:** [3.2ft, 2.1ft]

These objects were accurately labeled and placed in the correct zone relative to the frame, providing meaningful spatial data to guide movement.

7.2 **Natural Language Guidance via SLM**

Once detection data is collected, the Small Language Model (SmolLM2 1.7B) processes object types, zones, and distances to generate contextually aware guidance. For example, in the above scenario, the output was:

*“Slightly move to the right. Path ahead is partially blocked by a person.”*

This instruction is concise yet descriptive, allowing a user to understand both the action and its justification. The language model ensures that each message aligns with the user’s real-time spatial context while maintaining simplicity for fast comprehension.

**7.3 Audio Optimization and Feedback Control**

Initially, users faced overlapping audio guidance, especially when multiple detection events occurred close together. This was resolved by introducing a delay mechanism where the system only speaks once every 90 frames (about every 3 seconds for 30 fps video). This ensures:

* Voice outputs are distinct and easy to follow.
* Instructions are delivered at a natural and non-intrusive rhythm.
* Reduced cognitive load for the user while maintaining safety.

**7.4 Performance and Responsiveness**

* **Frame Optimization:** Only every 30th frame is processed, reducing computational load while maintaining responsiveness.
* **Image Hashing:** Prevents redundant processing of visually similar frames, increasing speed and efficiency.
* **Real-time Simulation:** The system mimics real-world reaction timing, which is essential for mobility assistance applications.

7.5 **System Evaluation & Observations**

While formal benchmarks (e.g., BERT or BLEU scores) were not calculated in this PoC stage, subjective evaluations and logs suggest:

* **YOLO12n Accuracy:** Robust object detection, even in slightly blurry or motion-heavy frames.
* **SLM Output Quality:** Generated guidance is consistent, interpretable, and adaptive to varying object configurations.
* **Latency:** The delay between frame capture and instruction generation was low, making the system suitable for real-time use.
* **Robustness in Indoor Environments:** Tests in corridors and rooms with furniture showed the system can handle typical indoor obstacles.

**7.6 User-Centered Impact**

The greatest value of this system lies in its real-world application. For individuals who are blind or visually impaired, navigating unfamiliar or cluttered spaces poses serious safety challenges. This system offers:

* Real-time awareness of surroundings through spoken feedback.
* Reduced anxiety while moving through environments.
* Autonomy in decision-making based on understandable cues.

In essence, this tool can be a life-enhancing aid, bridging the gap between the physical environment and non-visual interpretation.

**CHAPTER 8: Conclusion**

**8.1 Recap of the Work**

This project involved designing and implementing a fully offline, real-time guidance system using video input. By integrating:

* **YOLO12n** for lightweight object detection,
* **SmolLM2 1.7B** for natural-language navigation guidance,
* **Image hashing** for efficient frame filtering,
* and **distance estimation** using a camera-based calculation model,

we created a complete pipeline capable of understanding surroundings and translating them into voice-guided instructions. The system was built with a strong focus on performance, clarity, and user safety.

**8.2 Key Outcomes**

* **Real-Time Navigation:** Achieved low-latency detection and instruction generation suitable for real-world use.
* **Accurate Object Localization:** Objects were correctly identified and spatially categorized into Left, Center, and Right zones.
* **Dynamic Instruction Logic:** Advanced navigation decision-making (e.g., "Slightly Right") helps refine path suggestions in complex environments.
* **Natural Language Feedback:** The small language model was able to produce simple, context-aware instructions without internet dependency.
* **Speech Optimization:** Audio guidance was refined using controlled timing intervals to prevent overlap and improve clarity.

**8.3 Benefits of This Approach**

* **Offline Functionality:** The entire system works without needing cloud APIs or internet, making it highly suitable for low-resource or remote settings.
* **Edge-Device Friendly:** Utilization of YOLO12n and SmolLM2 ensures compatibility with mobile or embedded devices.
* **User-Centric Design:** Spoken feedback mimics human-like assistance, offering clarity and comfort to the user.
* **Scalable & Extendable:** The architecture can be enhanced with GPS, additional sensors, or gesture controls for broader applications.

This solution demonstrates how AI can be meaningfully adapted to assistive use cases, ultimately empowering users with enhanced independence and spatial awareness. Future enhancements could include obstacle tracking, user feedback loops, or integration with wearable hardware for deployment at scale.

**CHAPTER 9: Future Enhancements**

**1. Mobile Hardware Integration**

The next logical step is porting the entire system to run on mobile devices (Android/iOS) or compact embedded platforms (like Raspberry Pi or Jetson Nano). This transition will make the system wearable and portable, increasing its accessibility and daily usability for visually impaired users.

**2. Enhanced Voice Interaction**

Future versions could support:

* **Two-way voice interaction** (user queries and system responses).
* **Multi-language support** to cater to a broader demographic.
* **Voice-triggered controls**, allowing users to initiate or pause navigation through speech commands.

**3. GPS and Geolocation Integration**

Integrating GPS would allow the system to:

* Provide **location-aware guidance**, useful in outdoor environments.
* Alert users about nearby landmarks or road crossings.
* Combine with navigation maps to offer **end-to-end route guidance**.

**4. Real-World Testing with Volunteers**

The current system has been validated in simulated and pre-recorded environments. Testing with actual users will help:

* Understand practical usability.
* Gather valuable feedback on instruction clarity, timing, and comfort.
* Discover unforeseen edge cases in daily scenarios (e.g., crowded areas, poor lighting).

**5. Obstacle Tracking & Path Prediction**

Future iterations may incorporate:

* **Temporal tracking** of moving objects.
* **Path prediction algorithms** to identify evolving threats.
* More advanced logic for **dynamic decision-making** in changing environments.

**6. Modular Expansion**

With the system’s modular design, it can be extended to include:

* **Haptic feedback** (vibration alerts for deaf-blind users).
* **Cloud sync** for usage analytics (if internet is permitted).
* **Integration with smart glasses** or wearable cameras.

**CHAPTER 10: References**

1. Bochkovskiy, A., Wang, C.Y., & Liao, H.Y.M. (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection*. arXiv:2004.10934.
2. Ultralytics. (2023). *YOLO12n Documentation*. Retrieved from <https://docs.ultralytics.com/models/yolo12/>
3. Hugging Face. (2024). *SmolLM2-1.7B Model Card*. Retrieved from [https://huggingface.co/HuggingFaceTB/SmolLM2-1.7B-Instruct](https://huggingface.co/HuggingFaceTB/SmolLM2-1.7B-Instruct%20)
4. Redmon, J., & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement*. arXiv:1804.02767.
5. OpenCV Team. (2023). *OpenCV Documentation*. Retrieved from <https://docs.opencv.org>
6. Paszke, A., et al. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. NeurIPS.
7. Simonyan, K., & Zisserman, A. (2015). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. arXiv:1409.1556.
8. Szegedy, C., et al. (2015). *Going Deeper with Convolutions*. CVPR.
9. Lin, T.Y., et al. (2014). *Microsoft COCO: Common Objects in Context*. ECCV.
10. Wang, C.Y., Bochkovskiy, A., & Liao, H.Y.M. (2022). *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. arXiv:2207.02696.
11. Ruder, S. (2019). *Transfer Learning in NLP*. Retrieved from <https://ruder.io/nlp-transfer-learning>
12. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
13. Russakovsky, O., et al. (2015). *ImageNet Large Scale Visual Recognition Challenge*. IJCV.
14. LeCun, Y., Bengio, Y., & Hinton, G. (2015). *Deep learning*. Nature, 521(7553), 436-444.
15. GitHub. (2023). *YOLOv8 GitHub Repository*. Retrieved from <https://github.com/ultralytics/ultralytics>
16. Gfeller, B., et al. (2023). *Efficient and Compact Language Models*. Google Research. Retrieved from <https://ai.googleblog.com>
17. TensorFlow. (2024). *Object Detection API*. Retrieved from <https://tensorflow.org/lite/models/object_detection>
18. Pydub Documentation. (2023). *Audio Handling in Python*. Retrieved from <https://pydub.com>
19. Tellez, E. S., et al. (2019). *A New Approach for Blind Navigation Based on Deep Learning and Semantic Segmentation*. Sensors.
20. IEEE Xplore. (2020). *Assistive Technologies for the Visually Impaired: A Review*. Retrieved from <https://ieeexplore.ieee.org>

**CHAPTER: 11 Appendix**

**11.1 Sample Code Snippets**

Figure 5:

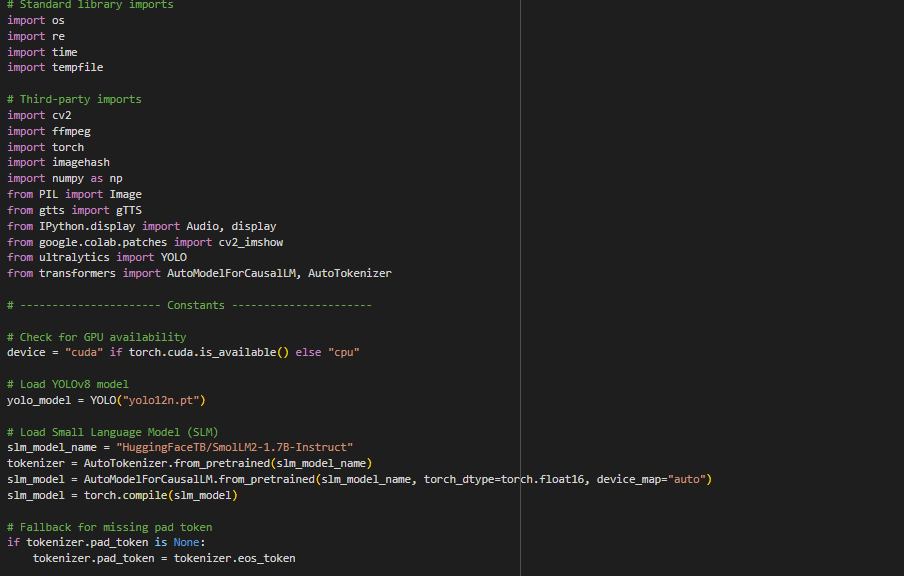
****

Figure 6:

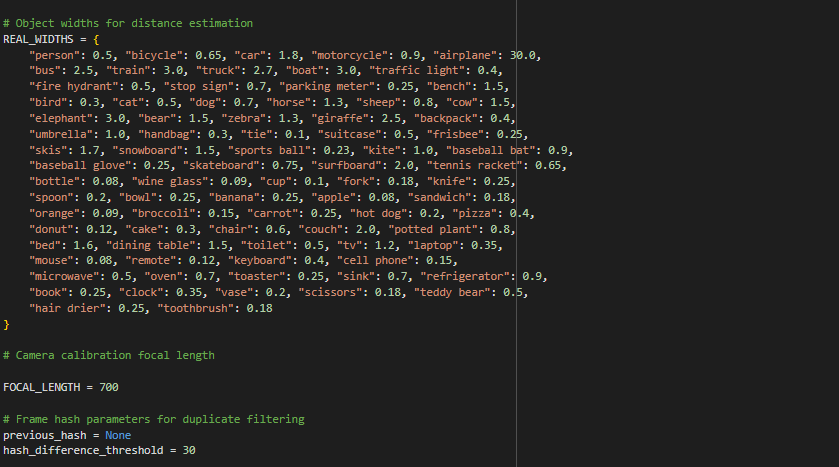
****

Figure 7**:**

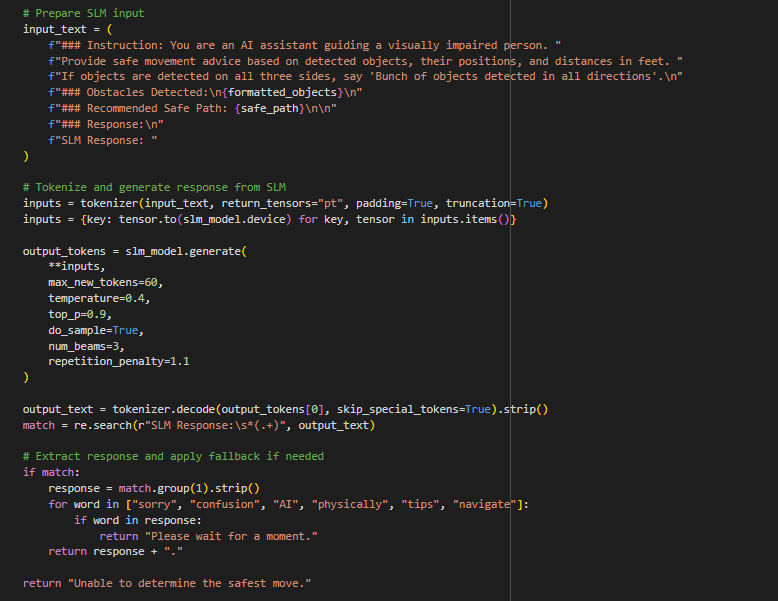
****

Figure 8:

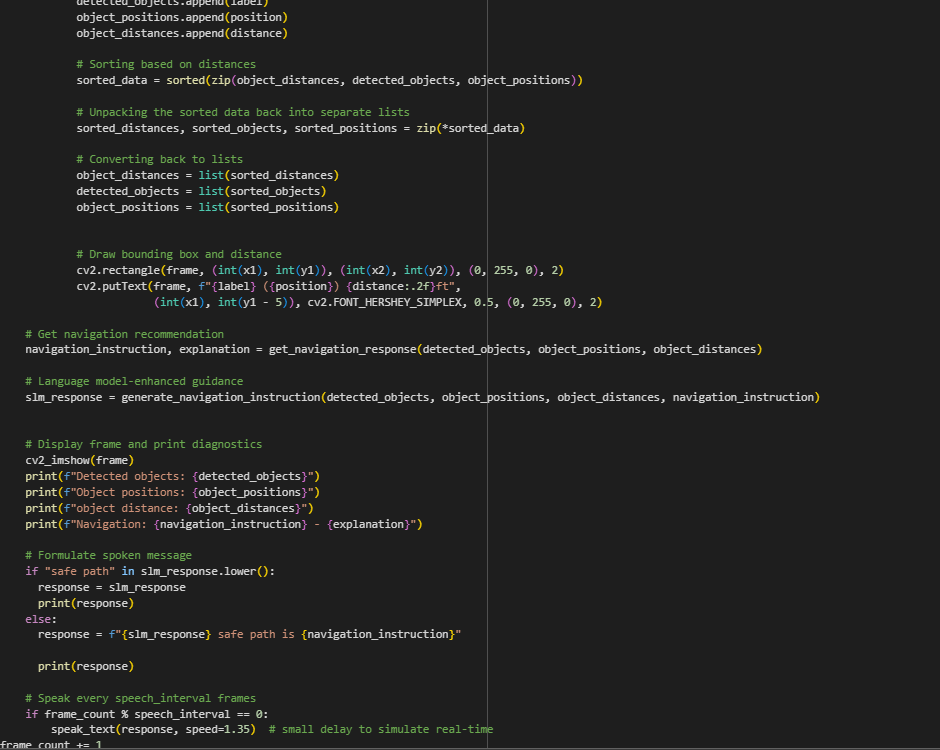
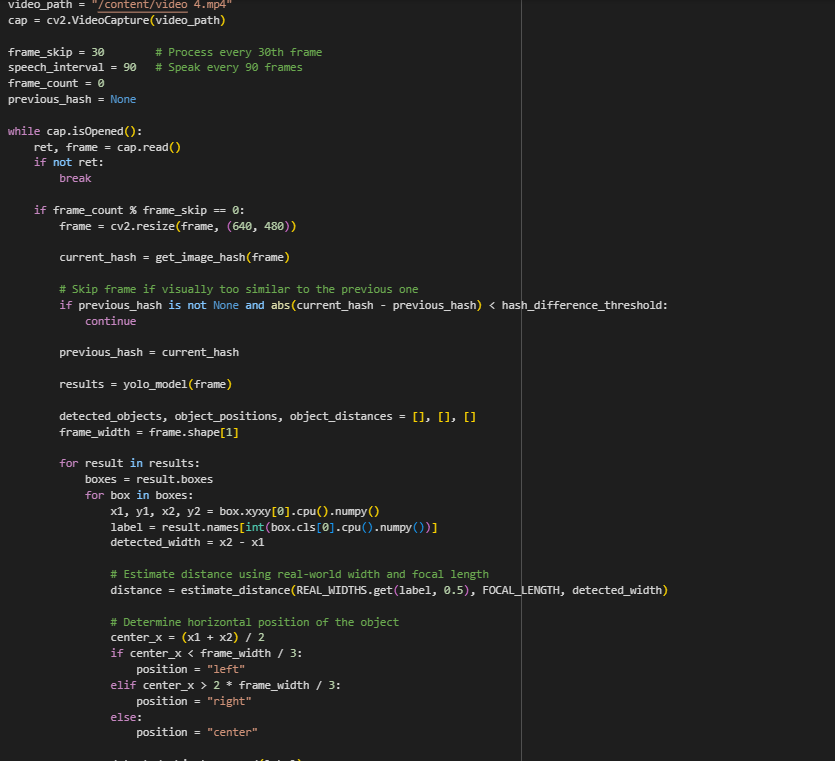
****

Figure 9:

****

**11.2 OUTPUT PICTURES :**

Figure 10:



Figure 11:

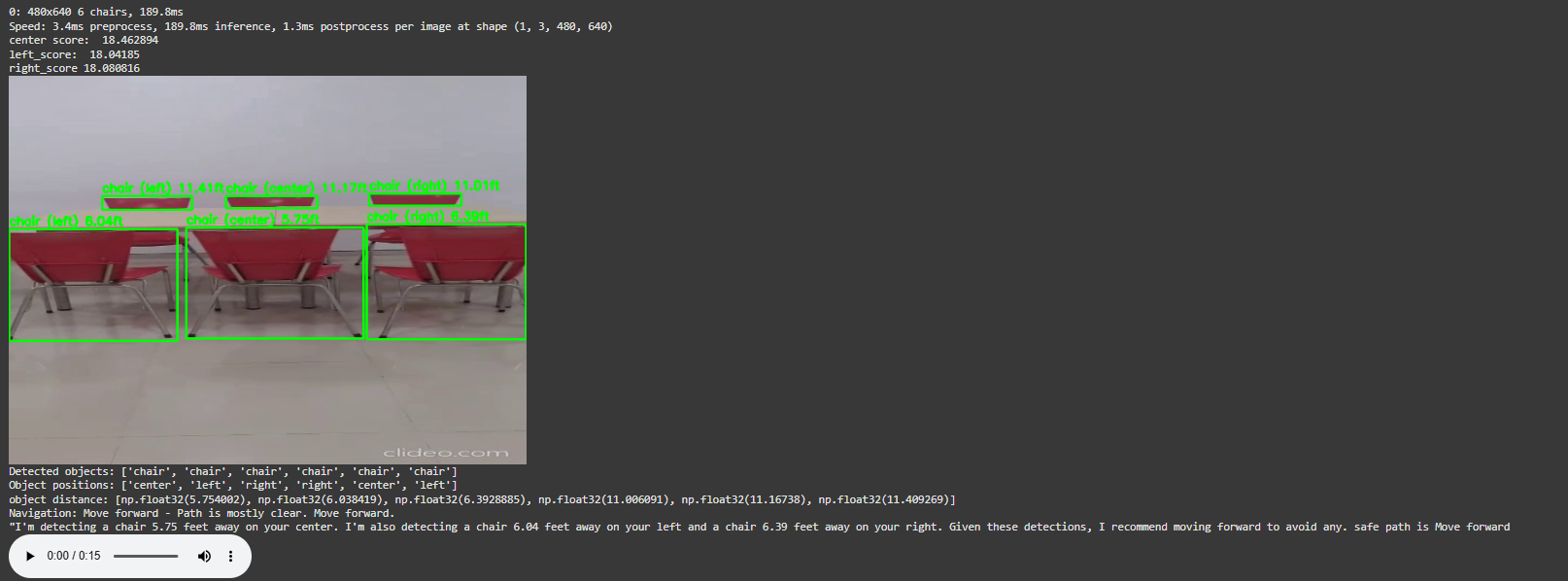


Figure 12:



Figure 13:



Figure 14:



Figure 15:

