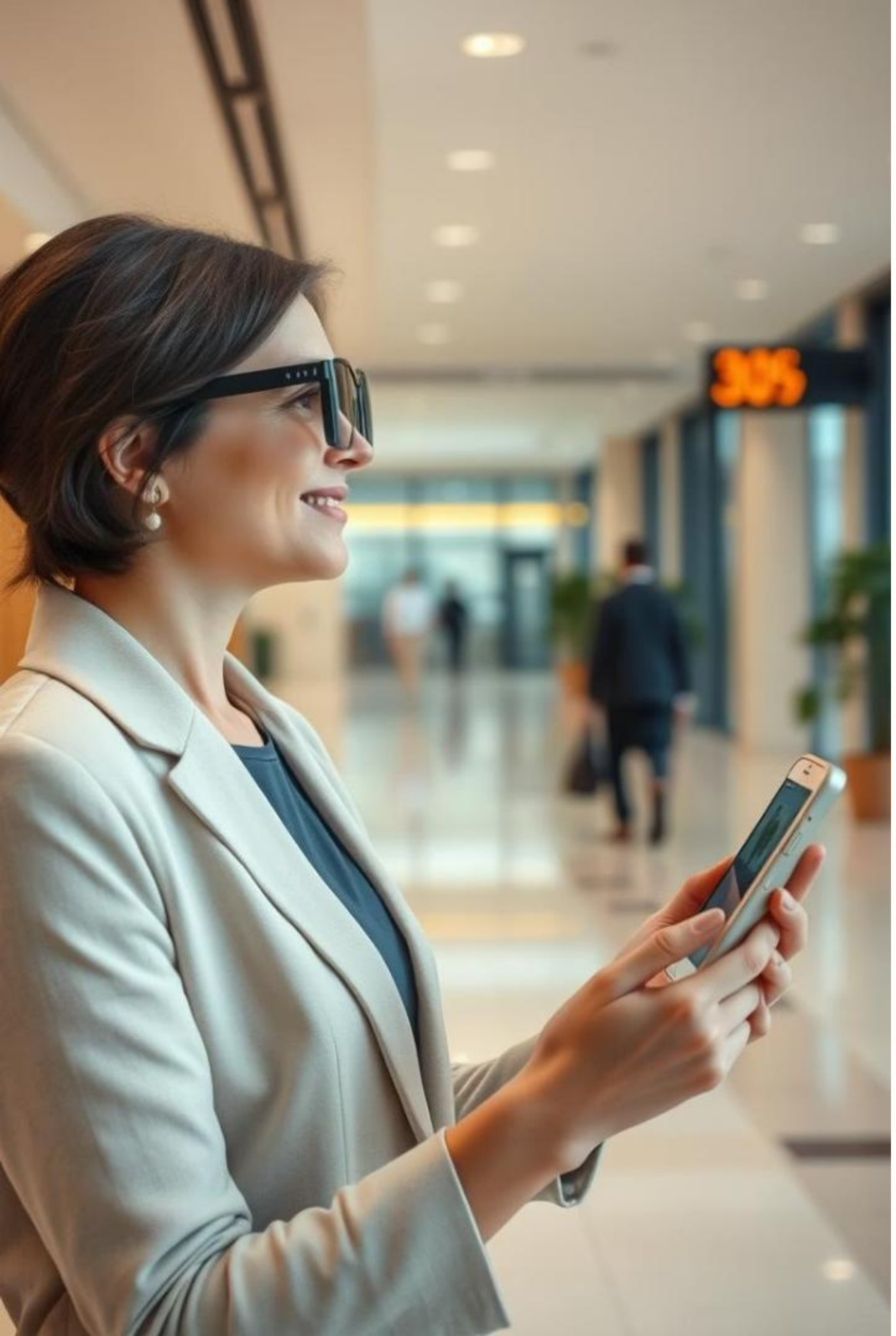


# Indoor Navigation for the Blind: Real-Time Object Detection and Mapping

Navigational assistance for visually impaired individuals often faces limitations in cost and indoor effectiveness. We introduce a real-time assistive system using a client-server architecture accessible via smartphone. The system integrates YOLOv5 object detection, MiDaS depth estimation, and ORB visual odometry for robust scene understanding. This approach provides an accessible, infrastructure-free navigation solution at interactive frame rates.

INDRAPRASTHA INSTITUTE *of* INFORMATION TECHNOLOGY **DELHI**

## Introduction to Indoor Navigation Challenges

### GPS Limitations

Indoor navigation poses unique challenges due to GPS signal limitations and dynamic environments.

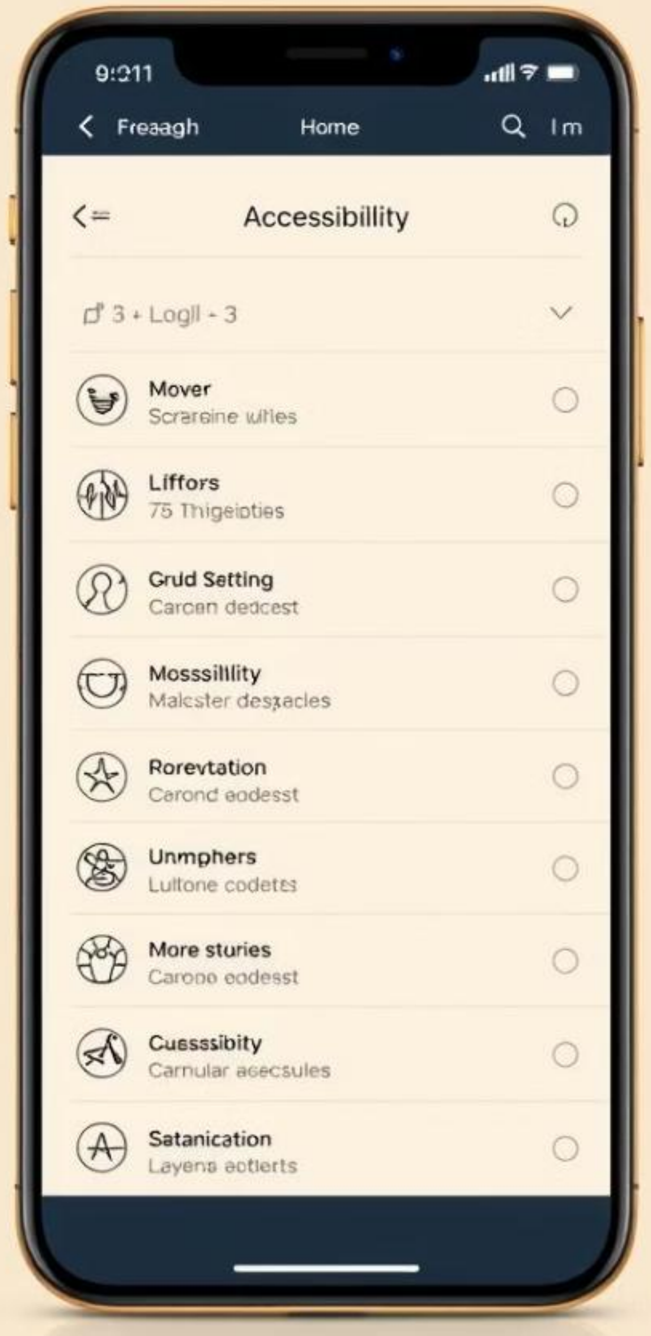
### Traditional Aids

Traditional aids like white canes provide insufficient spatial awareness without semantic understanding of surroundings.

### Mobile Application

This project presents a mobile application that employs computer vision for assistive indoor navigation.

The system integrates optimized deep learning models to deliver real-time environmental interpretation. Path planning is handled through the A* algorithm, ensuring safe navigation routes. The application features an Android frontend built with Kotlin and a scalable FastAPI backend containerized using Docker.



# Motivation Behind the Project

## Accessibility

Providing an affordable, camera-only solution that leverages existing smartphone or wearable hardware.

## Scalability

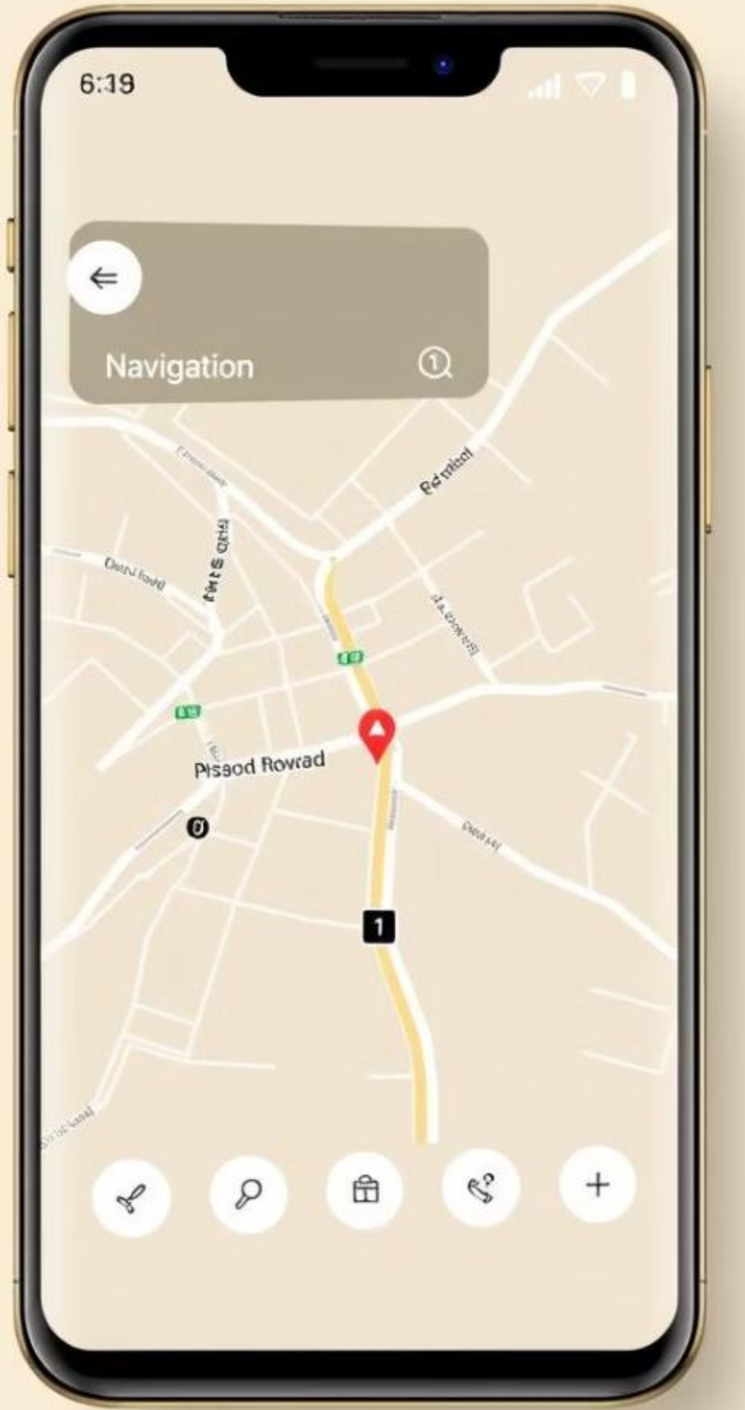
Eliminating the need for infrastructure to enable deployment in diverse indoor settings.

## Real-time Guidance

Ensuring low-latency perception and instruction so users receive timely alerts.

The project prioritizes the needs of visually impaired users by delivering intuitive, non-visual cues that translate complex spatial information into simple, actionable guidance, thereby enhancing autonomy and reducing cognitive load.

INDRAPRASTHA INSTITUTE *of* INFORMATION TECHNOLOGY DELHI



# Key Project Outcomes

## Android Application

Developed a user-friendly Android application using Kotlin, enabling real-time interaction and feedback.

## Lightweight Deep Learning Model

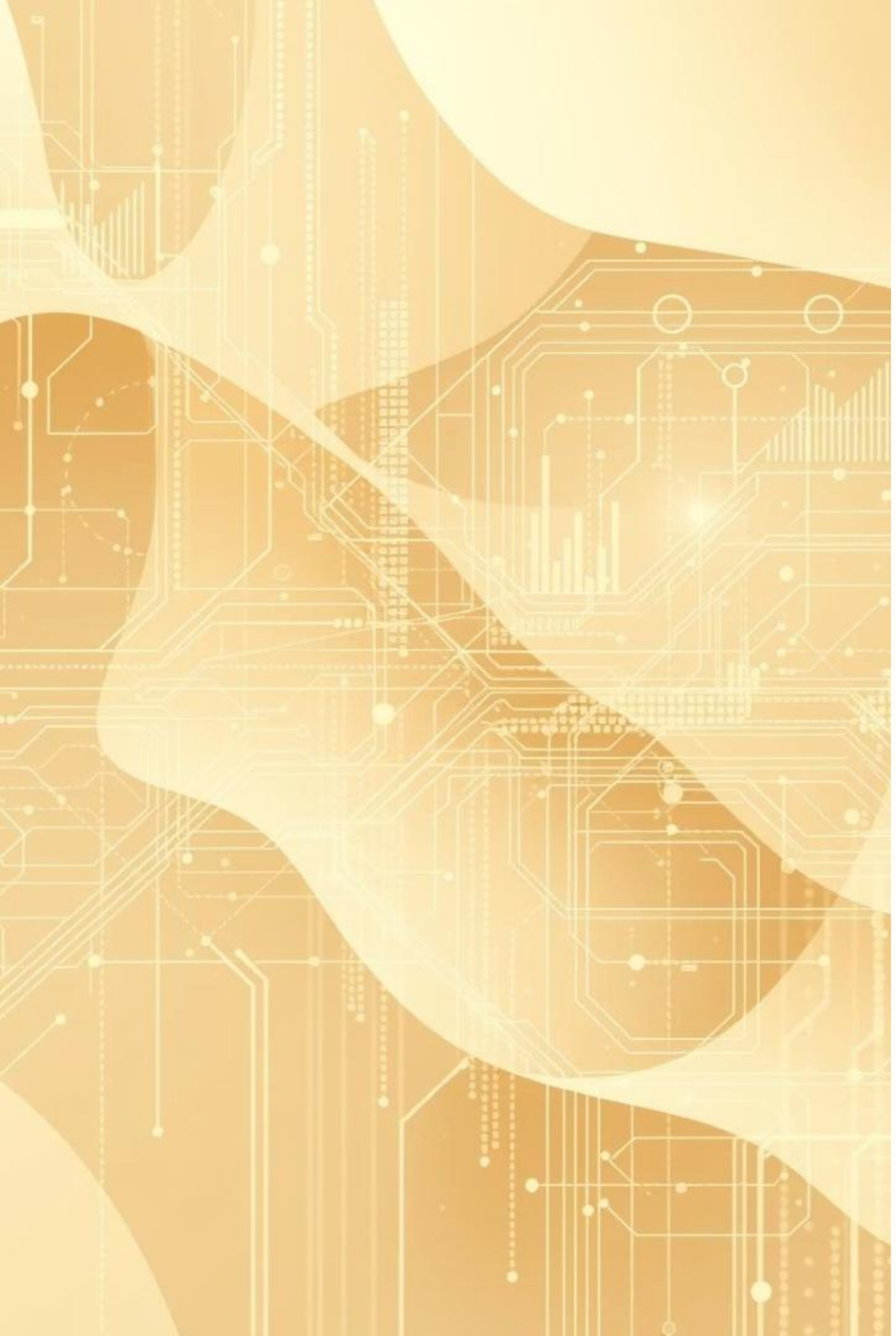
Implemented a lightweight deep learning model optimized for mobile devices, combining YOLO for object detection and MiDaS for depth estimation.

## Efficient Path Planning

Integrated the A* algorithm for efficient path planning, allowing the system to compute optimal routes while avoiding obstacles.

Designed a robust backend using FastAPI, containerized with Docker, ensuring scalability and ease of deployment. Conducted comprehensive testing in various indoor environments to validate the system's effectiveness and reliability.

INDRAPRASTHA INSTITUTE *of* INFORMATION TECHNOLOGY **DELHI**



## Addressing Core Challenges

1

### Spatial Variability

Indoor spaces exhibit diverse layouts, textures, and lighting conditions.

2

### GPS-Denied Localization

Vision-based SLAM must provide drift-free pose estimates over long trajectories.

3

### Depth Estimation

Monocular and stereo depth cues are often noisy under low-texture or poor lighting.

4

### Real-Time Constraints

Running object detection and depth estimation at interactive frame rates on mobile GPUs requires highly optimized inference.

Addressing these challenges is critical to realizing a vision-only indoor navigation system that delivers accurate, low-latency guidance to visually impaired users in real-world settings.

INDRAPRASTHA INSTITUTE *of* INFORMATION TECHNOLOGY DELHI

# Monocular Visual Odometry (Literature Review)

### Early SLAM Framework

Davison introduced a real-time monocular SLAM framework showing a single moving camera can estimate pose and map surroundings without external sensors.

### Our Approach

Building on this, we adopt lightweight monocular visual odometry using OpenCV-based feature tracking for continuous localization and navigation.

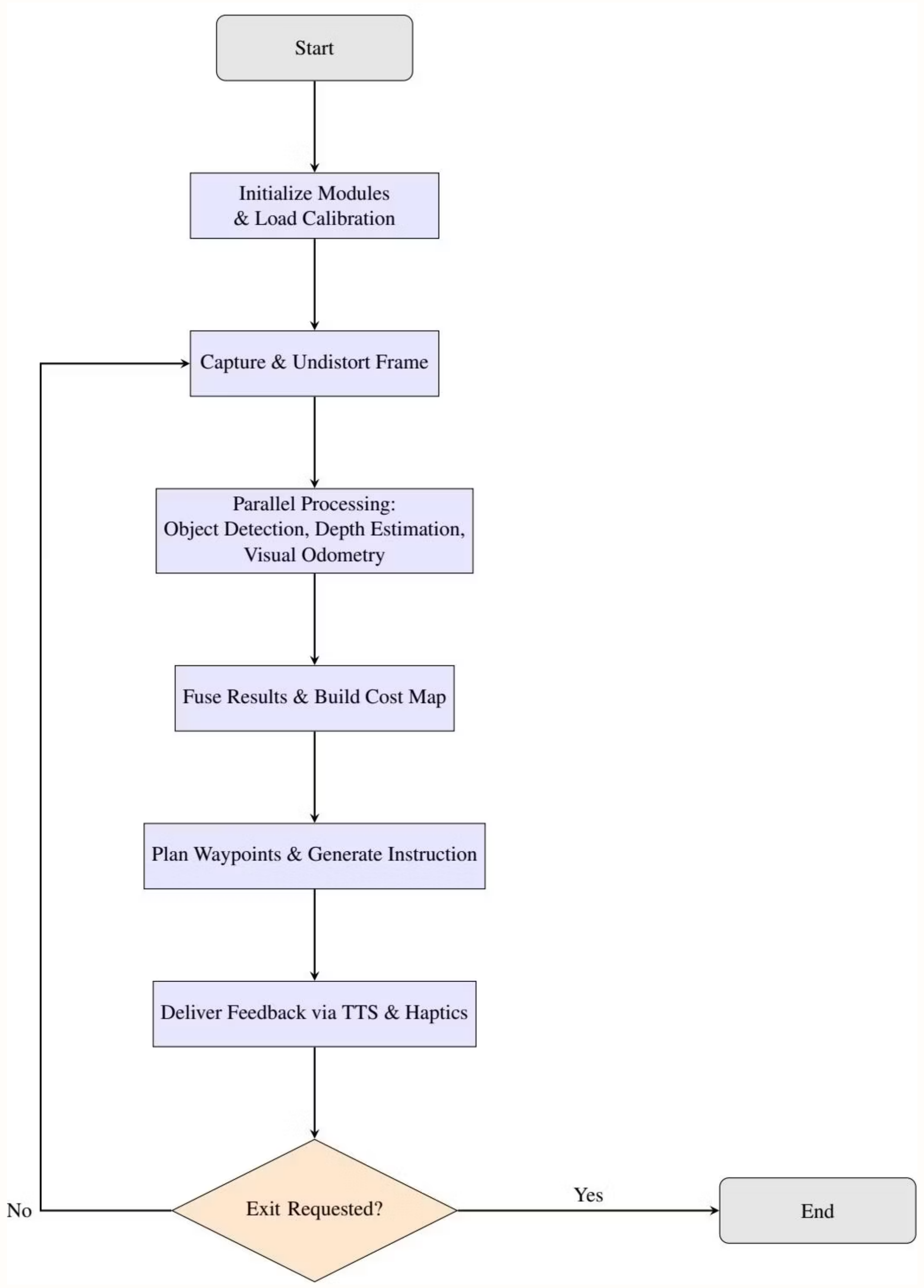
### Assistive Technology Focus

This approach enables scalable, infrastructure-free visual assistance critical for the visually impaired.

## Main Pipeline Overview

Bow we cover the architecture and implementation of our main pipeline for real-time navigation and object detection. We will explore the computer vision module, trajectory planning, instruction generation, and performance metrics. Our team collaboratively developed this system integrating vision, navigation, and audio feedback.

## Computer Vision Module Architecture


Start
Initialize Modules & Load Calibration
Capture & Undistort Frame
Parallel Processing: Object Detection, Depth Estimation, Visual Odometry
Fuse Results & Build Cost Map
Plan Waypoints & Generate Instruction
Deliver Feedback via TTS & Haptics
Exit Requested?
No
Yes
End

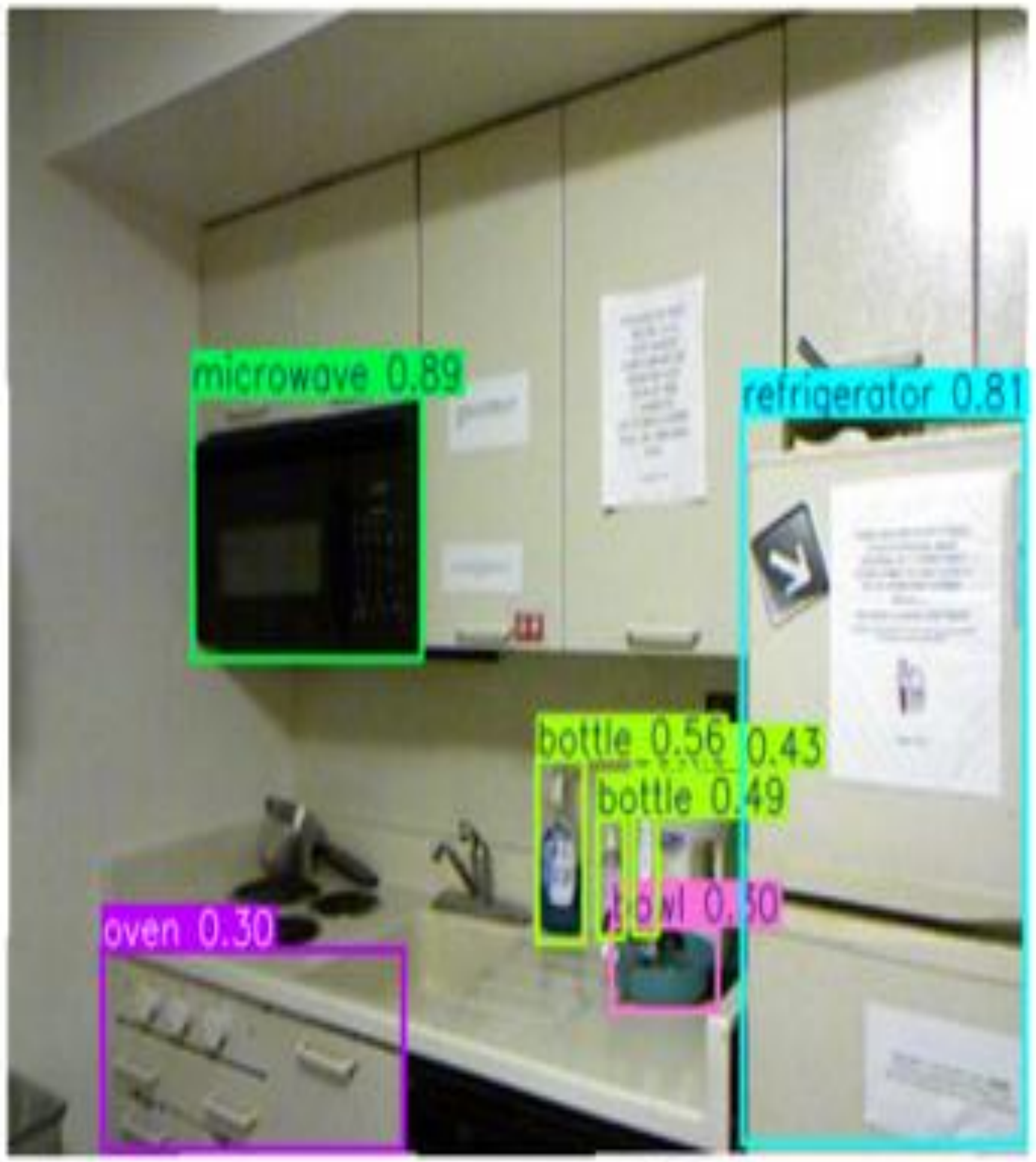
### Parallel Pipelines

The CV module runs three parallel pipelines using CUDA streams: object detection, depth estimation, and visual odometry. This design enables efficient processing of RGB frames for real-time analysis.

### Output Data

Outputs include bounding boxes, class labels, confidence scores, depth maps, and 2D trajectory data. These feed into the navigation and planning modules for obstacle avoidance and pathfinding.

YOLOv8 Detections



# Object Detection with YOLOv5s

## Model Details

Uses pretrained YOLOv5s on COCO dataset, processing 640×480 RGB frames.

## Detection Criteria

Retains detections with confidence > 0.4 and applies NMS with IoU threshold 0.45 to remove redundant boxes.

## Performance

Inference latency is approximately 12 ms per frame, enabling fast detection.

INDRAPRASTHA INSTITUTE *of* INFORMATION TECHNOLOGY DELHI

# Depth Estimation Pipeline

## Model and Processing

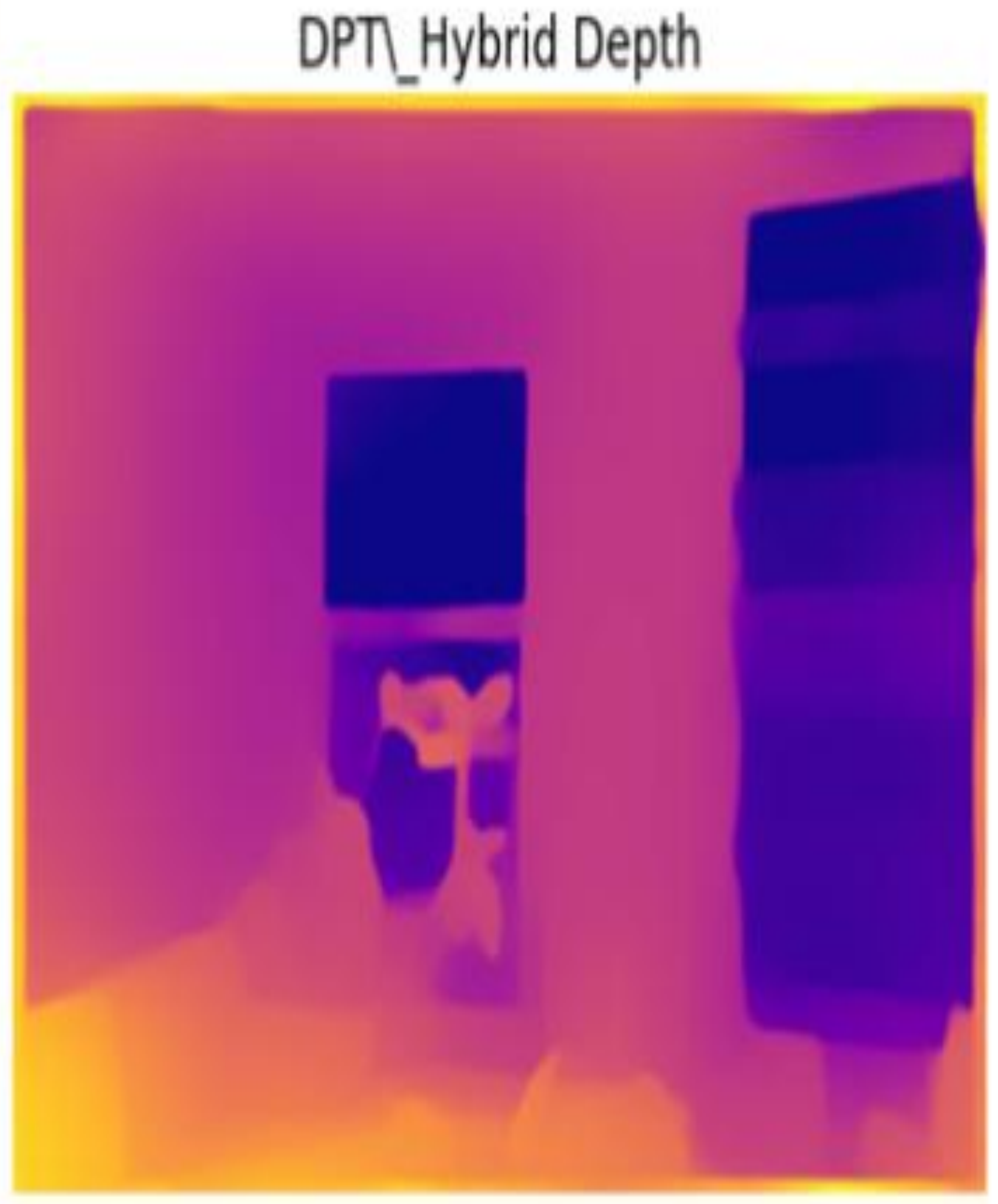
MiDaS DPT Hybrid model processes resized 384×384 RGB frames to generate depth maps.

## Filtering and Normalization

Depth maps are filtered with median and bilateral filters, then normalized to a closeness map indicating object proximity.

## Latency and Resolution

Depth estimation latency is about 18 ms per frame, with upsampling to original frame resolution for accuracy.



# Visual Odometry and Trajectory Estimation

### Feature Detection

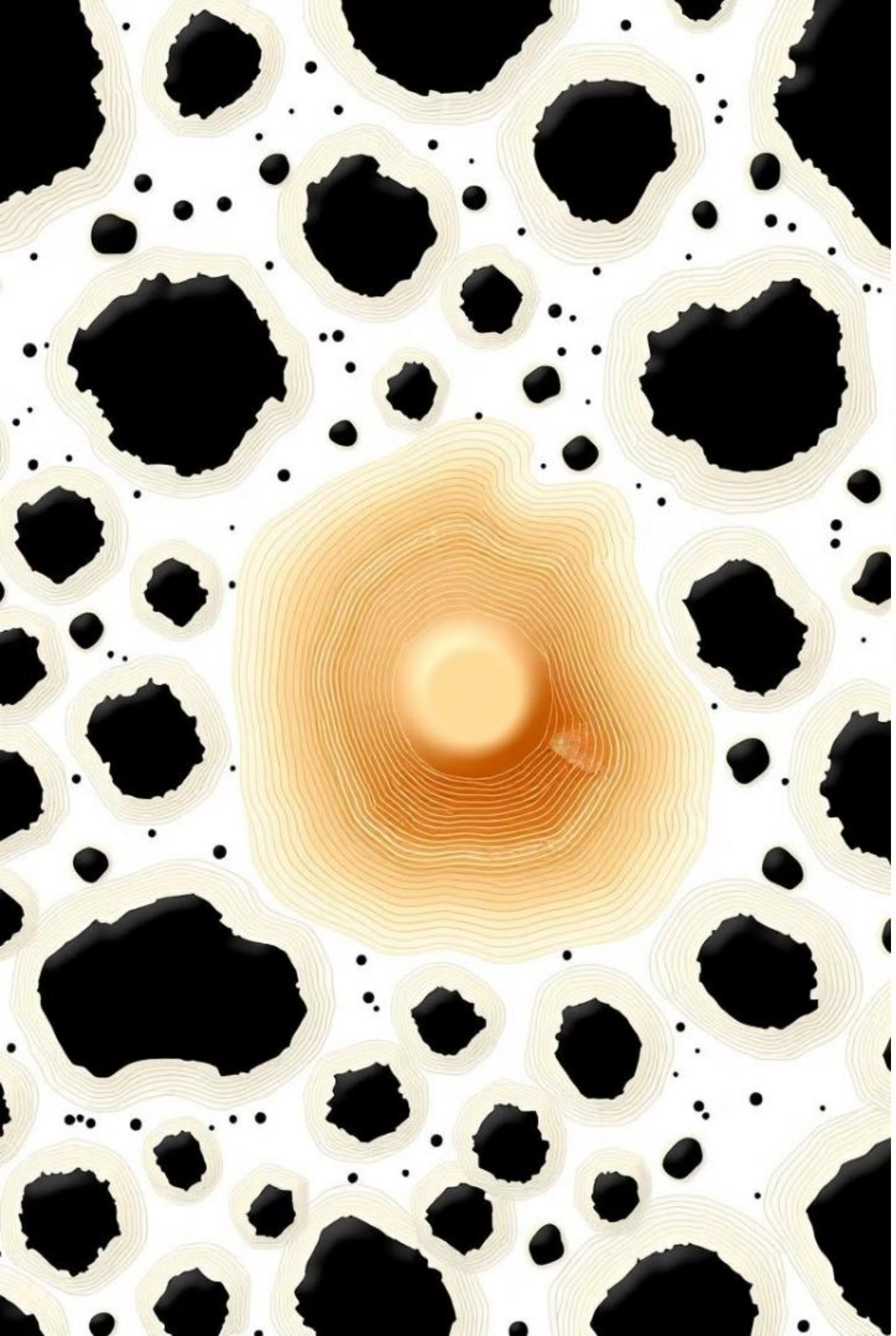
Detects up to 1500 ORB features on grayscale frames, matching across frames using Hamming distance.

### Pose Estimation

Uses Perspective-n-Point (PnP) to estimate relative pose when depth map is available; otherwise, pose updates are skipped.

### Trajectory Construction

Accumulates poses to build a 2D trajectory in pixel coordinates for navigation.



# Fusion and Obstacle Mapping

## Obstacle Identification

Samples closeness map at object centers; pixels with closeness > 0.5 are marked as obstacles.

## Map Processing

Obstacle map is dilated by 15 pixels and distance transform computes Euclidean distances for navigation cost.

## Navigation Cost

Cost combines proximity and closeness weighted by factors wprox=300 and wclose=50, guiding path planning.

# Trajectory Planning with A* Algorithm

## Grid and Steps

Planning occurs on an 8-connected grid with 15-pixel step sizes for waypoint generation.

## Cost Function

Cost combines navigation cost and heuristic distance to goal, with heuristic weight wh=1.5.

## Termination Criteria

Planning ends when current point is within 1.5 steps (22.5 pixels) of the goal, producing a waypoint sequence.

# Instruction Generation and Text-to-Speech

## Instruction Types

Waypoint deltas convert to heading changes classified as forward, slight left/right, or turn left/right based on angle thresholds.

## Distance Accumulation

Distances are accumulated in 15-pixel units to provide meaningful navigation cues.

## Audio Feedback

A dedicated TTSManager thread uses pyttsx3 to speak instructions non-blockingly, ensuring real-time audio guidance.

# Implementation and Performance Metrics

## Software Stack

Implemented in Python using OpenCV, PyTorch, NumPy, and pyttsx3 with parallelism via concurrent.futures.

## Performance

Achieves approximately 25 FPS end-to-end on RTX 3080 and i7 CPU, with module latencies: YOLOv5s 12 ms, MiDaS 18 ms, VO 10 ms, planning & TTS 5 ms.

## Demo Video

Refer to the demo video on Google Drive for a visual demonstration of the CV pipeline in action.

# Experimental Metrics and Compute Requirements

## Object Detection

mAP@0.50: 0.80, Precision: 0.56, Recall: 0.50, F1 Score: 0.60

## Depth Estimation

AbsRel: 1.2269 (MiDaS Small), 1.2840 (DPT Hybrid)

## Navigation

Success Rate: 71%, Collision Rate: 22%

The client-server architecture distributes computational load to optimize performance and battery life. The backend server requires a GPU, multi-core CPU, and 16GB+ RAM. The frontend mobile client requires a CPU, 4-8GB RAM, and a stable network connection.

# Backend Processing Engine Architecture

## Core Engine

Developed in Python using FastAPI, the backend efficiently receives image frames via HTTP POST from the client.

## Processing Steps

- Input validation and preprocessing
- Machine learning inference for object detection and spatial estimation
- Instruction generation for navigational cues
- Response formulation as structured JSON

## Frontend Mobile Application Features

### Camera Management

Captures live video feed continuously from the user's perspective.

### Network Communication

Encodes and sends camera frames to backend API, receives and parses JSON responses.

### Non-Visual Feedback

Delivers instructions via text-to-speech and haptic feedback for accessibility.

### User Interface

Accessible controls for starting/stopping assistance, adjusting settings, and optional live feed viewing.

INDRAPRASTHA INSTITUTE *of* INFORMATION TECHNOLOGY DELHI

## Implementation Technologies

### Backend Libraries

**FastAPI** for routing and async handling

**OpenCV** for image processing

**PyTorch** or TensorFlow for ML inference

### Frontend Tools

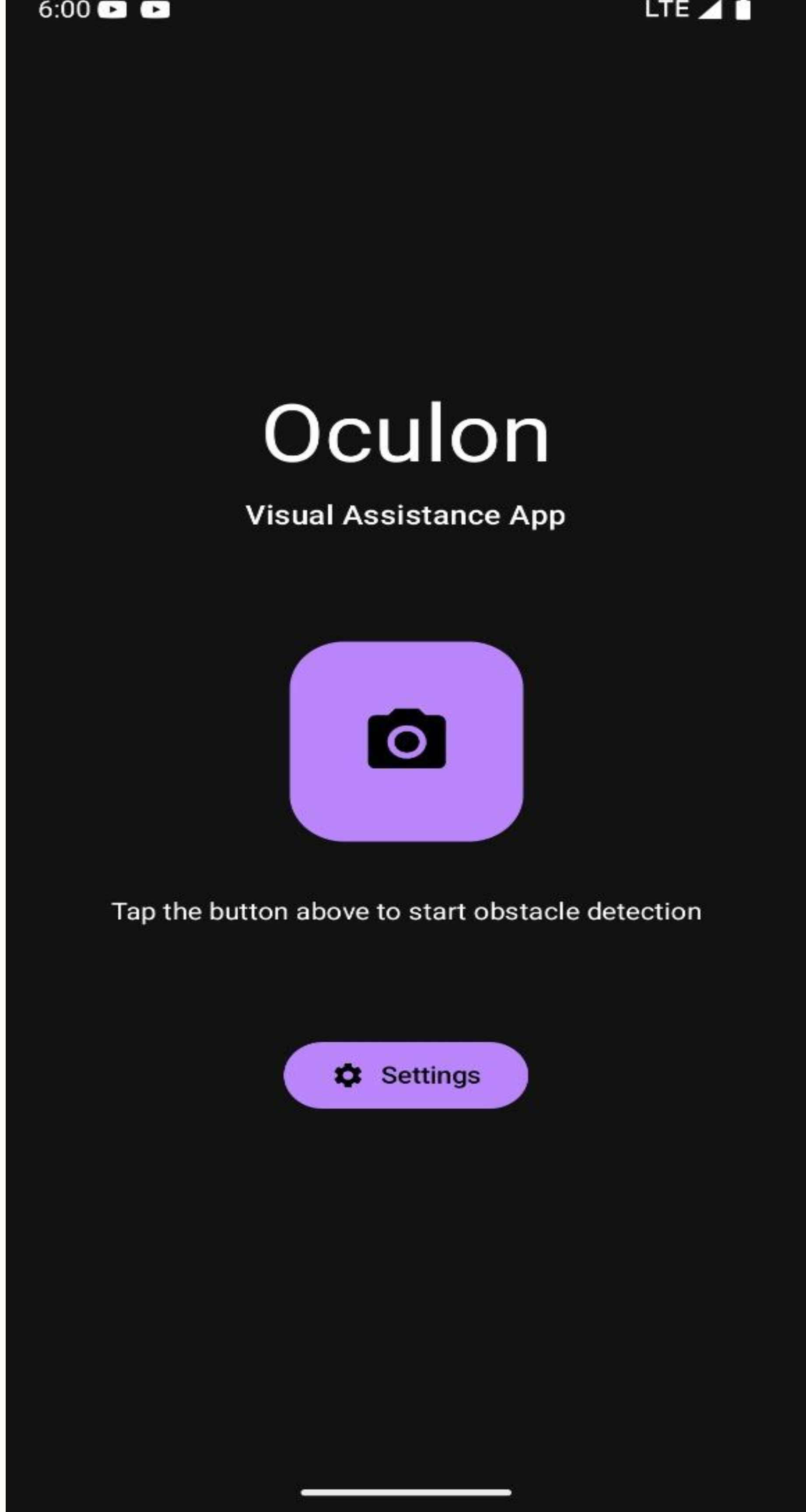
**Next.js (React)** for file-system based routing, server-side rendering (SSR), static site generation (SSG), and automatic code splitting

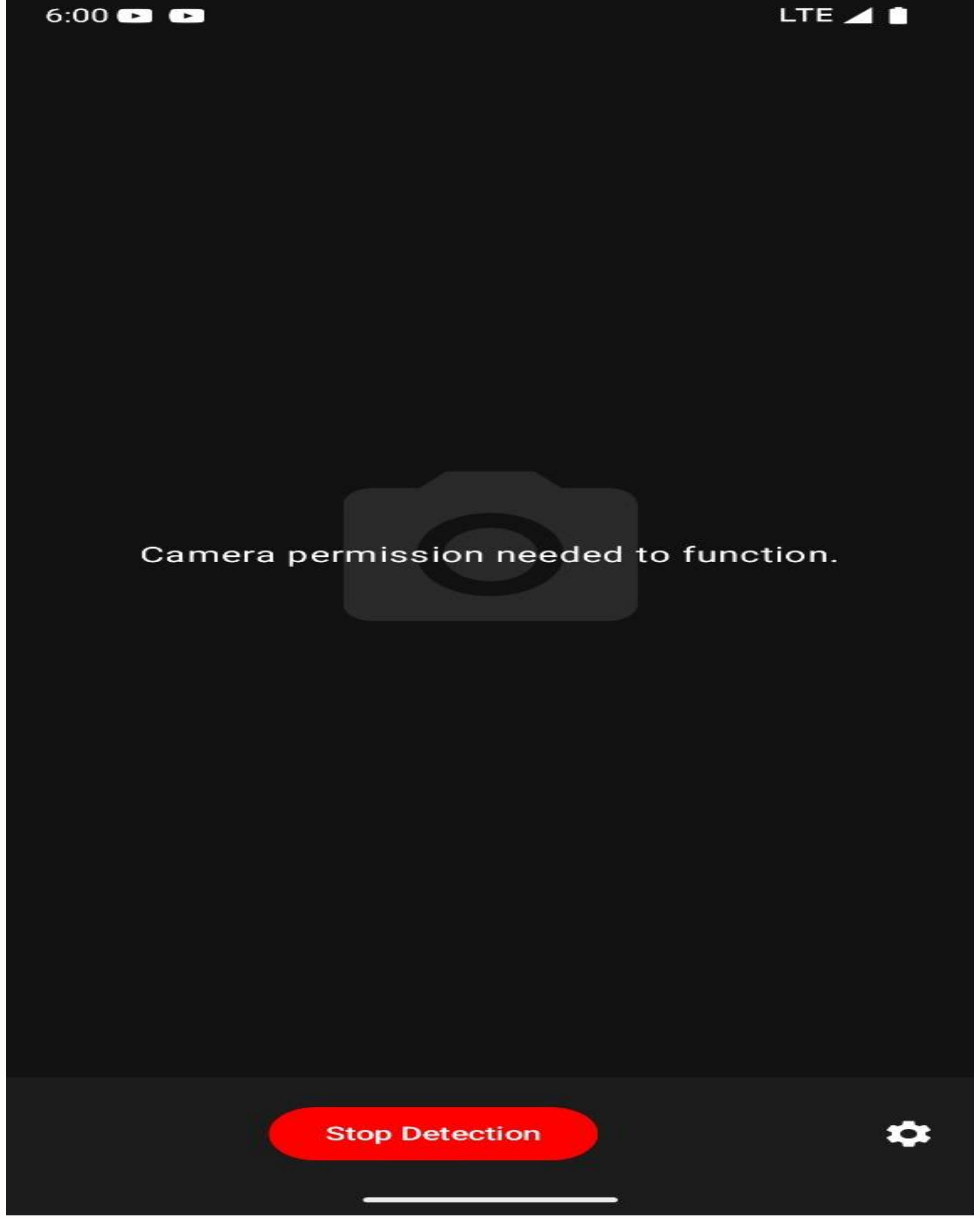
**JavaScript(ES6+)** with next/font for automatic optimization and self hosting  of Vercel's Geist font family.

**Vercel Platform** for zero-configuration deployment, global CDN delivery, and auto-scaled serverless functions

## System Workflow Overview

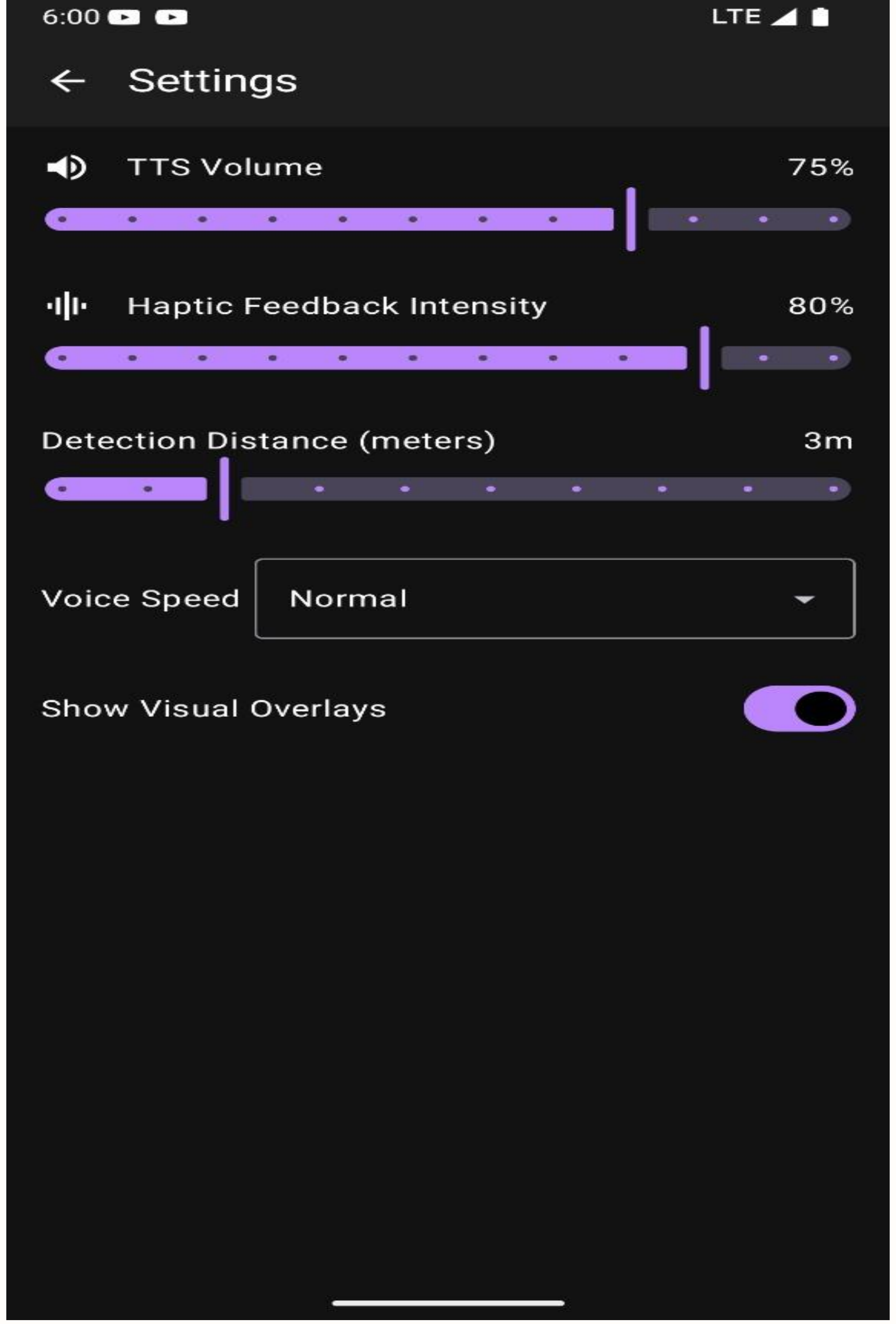
The real-time visual assistance pipeline starts with the client capturing video frames, which are sent to the backend for processing. The backend performs object detection and spatial analysis, then generates navigational instructions. These instructions are sent back to the client for delivery via accessible feedback modalities.





## Main Video Feed Page

Displays the live camera feed and allows users to start obstacle detection.



TTS Volume 75%

Haptic Feedback Intensity 80%

Detection Distance (meters) 3m

Voice Speed Normal

Show Visual Overlays

## Settings Page

Users can adjust TTS rate, haptic intensity, and other preferences.

INDRAPRASTHA INSTITUTE *of* INFORMATION TECHNOLOGY DELHI

# System Summary and Scalability

1

## Client-Server Architecture

Heavy computation is offloaded to a cloud backend accessed via a simple API.

2

## Efficient Frontend

Android client focuses on camera handling, network communication, and accessible feedback.

3

## Scalable Solution

Architecture supports diverse user needs and future enhancements in visual assistance technology.

# Future Work and Limitations

1

## Monocular Depth Noise

Ambiguity in texture-less regions, low lighting, and cluttered scenes.

2

## Lack of Absolute Scale

Up-to-scale depth estimates without direct metric distance understanding.

3

## Inference Latency

Performance fluctuations depending on scene complexity.

Future work includes improving depth estimation, scale disambiguation, and optimization. Advanced scene understanding and enhanced SLAM will transform the system from basic obstacle avoidance to comprehensive 3D semantic navigation.

## Team Contributions and References

### Team Effort

All team members contributed equally to the project, from vision module integration to real-time navigation and feedback systems.

### Key References

- Monoslam: Real-time single camera SLAM (Davison et al., 2007)
- Depth map prediction using multi-scale deep network (Eigen et al., 2014)
- Ultralytics YOLOv5 (Jocher, 2020)
- Deep learning-based positioning for visually impaired (Mahida et al., 2025)
- ORB-SLAM2 open-source system (Mur-Artal & Tardós, 2015)

INDRAPRASTHA INSTITUTE *of* INFORMATION TECHNOLOGY DELHI

# Conclusion

### Vision-Based

### Client-Server

### Autonomy

We present a real-time vision-based navigation system for visually impaired users that combines YOLOv5s, MiDaS, and visual odometry in a client-server architecture. Our system achieves interactive frame rates with reliable environmental understanding across varied conditions.