STATS 507 Project Proposal

Student Name: Akhil Nishad | UMID: 38818750 | Uniquename: akhilnis@umich.edu

"Describe My Environment" — Accessibility Tool for Low Vision Users

Overview

Background

Millions of blind and low-vision (BLV) individuals struggle to independently understand their visual surroundings in real-time. While traditional accessibility tools (magnification, screen readers) exist, they don't solve the core problem: **giving users rich, contextual descriptions of dynamic environments**.
Modern computer vision and natural language models now make it possible to automatically detect objects, understand scenes, and generate descriptive text in real-time. This project builds an **end-to-end accessibility application** that uses object detection and scene understanding to provide audio/text descriptions of a user's surroundings.

Why this Project?

This project addresses a real accessibility need while demonstrating practical application of modern computer vision and natural language processing. It combines multiple ML components (object detection, vision-language models, TTS) into an end-to-end system, showcasing full-stack ML engineering skills.

Data & Models

Models: YOLO11 (object detection), BLIP (HuggingFace vision-language model), SpeechT5/pyttsx3 (TTS)
Datasets: COCO (330K+ images, 80 object classes), Flickr30K (caption quality evaluation)

Expected Insights

- Performance benchmarks for real-time ML pipelines on consumer hardware
- Trade-offs between model complexity and latency for accessibility applications
- Effectiveness of combining object detection with vision-language models for scene understanding
- Practical challenges in deploying ML models for real-time accessibility tools

Prior Work

Literature review

Existing Commercial Solutions:

- **Microsoft Seeing AI** (2017): Mobile app using computer vision for object recognition and text reading. Proprietary, iOS-only.
- **Envision App** (2018): Commercial tool combining OCR and object detection. Subscription-based, limited customization.
- **EnVisionVR** (2021, University of Michigan): Research system using Vision Language Models (VLMs) for real-time scene descriptions in VR environments for BLV users. Demonstrates feasibility of VLM-based accessibility tools.

Recent Technical Advances:

- YOLO Series (Redmon et al., 2016-2024): Real-time object detection achieving >20 FPS with high accuracy. YOLO11 (2024) improves efficiency with 22% fewer parameters than YOLOv8.
- Vision-Language Models: BLIP (Li et al., 2022) and LLaVA (Liu et al., 2023) enable high quality image captioning. BLIP achieves 156ms latency on modern hardware, suitable for real-time applications.
- COCO Dataset (Lin et al., 2014): 330K+ images with 80 object categories, standard benchmark for object detection and captioning tasks

Other:

1. Ultralytics. YOLO11 Documentation. https://docs.ultralytics.com/
2. HuggingFace. Transformers: Vision-Language Models.
3. FreecodeAcademy: "How to Build AI Speech-to-Text and Text-to-Speech Accessibility Tools" (2025)

Gap: No open-source, end-to-end solution combining real-time object detection, scene understanding, and audio output for accessibility use cases.

Methods

Object Detection: YOLO11 (Ultralytics) pre-trained on COCO dataset provides real-time detection of 80 object classes with bounding boxes and confidence scores.

Scene Understanding: BLIP (HuggingFace Transformers) generates natural language descriptions from images. Alternative: LLaVA for richer captions (slower, requires more memory).

Text-to-Speech: pyttsx3 (system TTS) or SpeechT5 (HuggingFace) converts descriptions to audio with minimal latency.

Integration: Python backend (Flask/FastAPI) processes camera frames through detection → captioning → TTS pipeline. Next.js frontend provides an accessible web interface.

Technical Approach

Pipeline Architecture

```
┌─────────────────────────────────────────────────────────────┐
│                  USER INTERFACE (Next.js)                   │
│   - Live camera feed display                                │
│   - Real-time transcription & audio output                  │
│   - Interactive Q&A mode (ask about specific objects)       │
└─────────────────────────────────────────────────────────────┘
                            │
                            │
                            ↓ (WebSocket/REST API)
┌─────────────────────────────────────────────────────────────┐
│                 PYTHON BACKEND (Flask/FastAPI)              │
│                                                             │
│  ┌──────────────────┐  ┌──────────────────────────────────┐ │
│  │ Object Detection │  │  Scene Understanding & Captioning│ │
│  │ (YOLO11/PyTorch) │  │  (Vision-Language Model)         │ │
│  └──────────────────┘  └──────────────────────────────────┘ │
│             │                         │                     │
│             └─────────────────────────┘                     │
│                      ↓                                      │
│                                                             │
│               ┌─────────────────┐                           │
│               │  Text Generator │                           │
│               │  + Natural Lang.│                           │
│               └─────────────────┘                           │
│                       ↓                                     │
│               ┌──────────────────┐                          │
│               │ Text-to-Speech   │                          │
│               │ (SpeechT5/pyttsx)│                          │
│               └──────────────────┘                          │
│                       ↓                                     │
│              Output: Audio/Text                             │
└─────────────────────────────────────────────────────────────┘
```

Core Components

1. **Object Detection:** YOLO11 (PyTorch) detects 80 object classes from COCO dataset
2. **Scene Understanding:** BLIP vision-language model generates natural language descriptions
3. **Text-to-Speech:** pyttsx3 or SpeechT5 converts descriptions to audio
4. **Web Interface:** Next.js frontend with real-time camera feed and accessibility-first UI

Preliminary Results

Data Understanding

Initial experiments validate feasibility. Tests conducted on MacBook Pro M4 Pro (24GB Unified Memory) using PyTorch 2.9.1 with MPS backend.

<u>COCO Dataset</u>

- **Total Images:** ~330,000 images
- **Object Classes:** 80 categories
- **Stuff Classes:** 91 additional categories (ground, sky, etc.)
- **Train/Val/Test Split:** 118,287 / 5,000 / 40,670 images

Key Insights:

- Highly diverse scenes with cluttered environments ideal for accessibility tasks

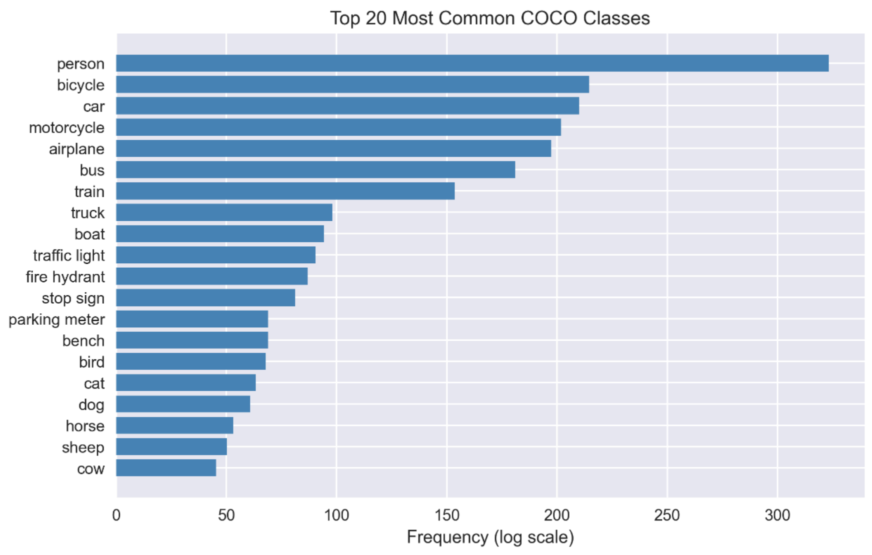
- Small-object detection presents challenges
- Images frequently include occlusion and multiple subjects
- Common objects (person, chair, car) appear frequently, while specialized objects are rarer

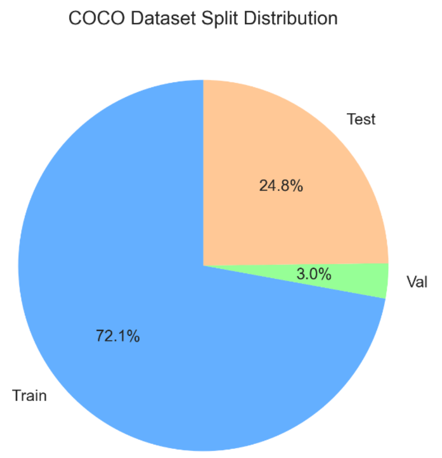
<u>**Flickr30K / COCO Captions**</u>

- **Images:** ~31,000 images with 5 captions each
- **Linguistic Variety:** Good diversity in caption styles and descriptions

Dataset Quality Notes:

- Captions sometimes subjective (different annotators describe scenes differently)
- Some scenes ambiguous without context
- Useful for training caption quality but may need filtering for accessibility use cases



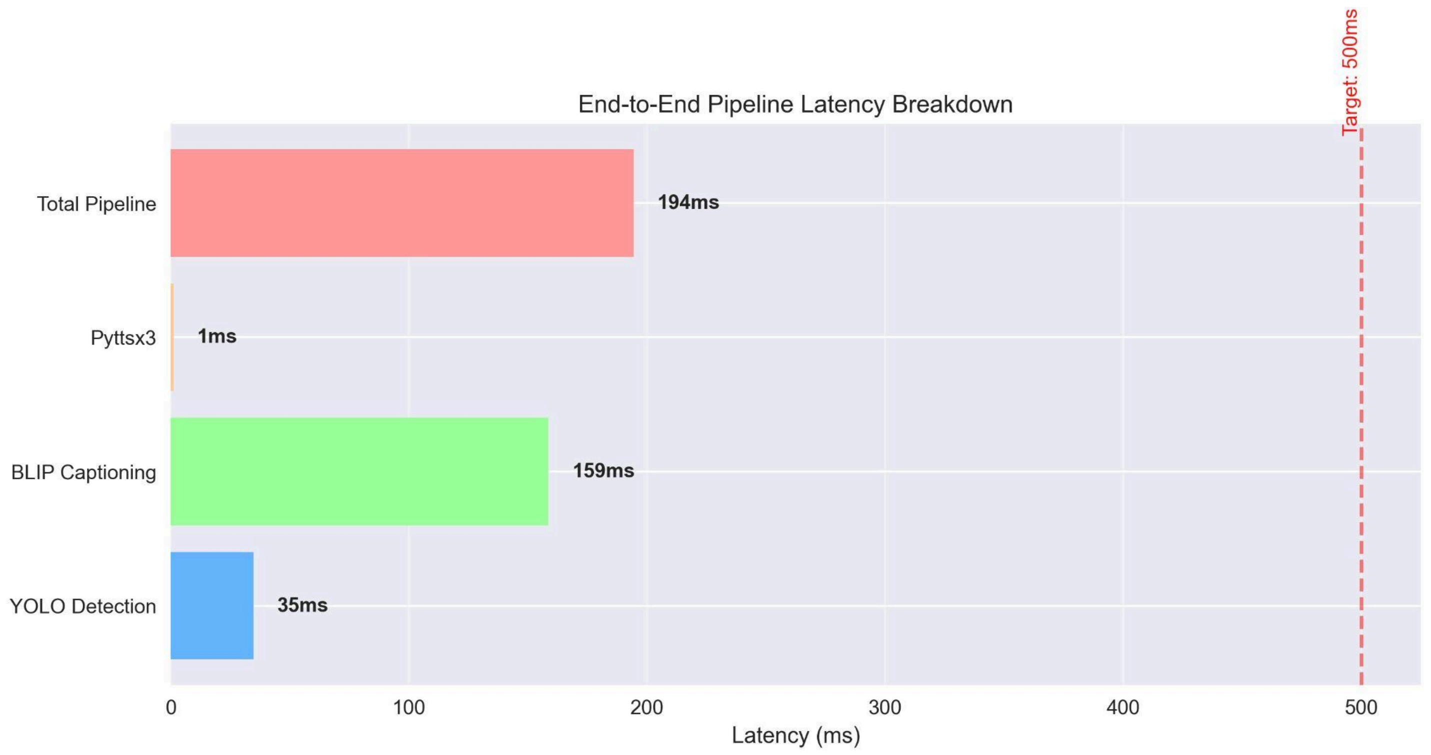


Model Performance & Resource Requirements

Initial experiments on a MacBook Pro M4 Pro (PyTorch 2.9.1, MPS) show that YOLO11n and YOLOv8n both achieve real-time object detection, with YOLO11n selected for its modern architecture and future support. Captioning tests with BLIP provide coherent scene descriptions at ~159 ms/image, while larger models like LLaVA are currently too memory-heavy for on-device real-time use but promising for richer, cloud-based interaction. For text-to-speech, pyttsx3 delivers sub-millisecond latency and stable performance, making it the preferred real-time TTS option for now, with SpeechT5 reserved for future exploration once dataset compatibility issues are resolved.

Benchmarks

Component	Observed Latency	Throughput	Memory Usage
YOLO11 Detection	34.7 ± 2.6 ms	28.8 FPS	~2GB
BLIP Captioning	158.7 ± 2.6 ms	6.3 FPS	~1.5GB
pyttsx3 TTS	1.1 ± 0.7 ms	N/A	Negligible
Total Pipeline	**194.4 ms**	**~5.14fps**	**~3.5GB**



Key Findings:

- BLIP performs 2× faster than expected (158.7ms vs 350-500ms literature)
- Real-time capability confirmed (exceeds 2 FPS target by 2.5×)
- TTS latency is negligible (<2ms), making it ideal for real-time applications
- System shows excellent promise for accessibility use cases

Performance Bottlenecks:

- Captioning latency (~82% of pipeline time) is primary bottleneck
- Model loading time (~2-5 seconds) requires warmup before real-time use
- Small object detection needs improvement for accessibility use cases

Tools Used & To Explore

Currently Used: NumPy, Pandas, Matplotlib/Seaborn, PyTorch, Transformers (HuggingFace), Jupyter Notebook

To Explore: COCO API (dataset evaluation), OpenCV (image preprocessing), FastAPI (backend API), Weights & Biases (experiment tracking)

Project Deliverables

Sub-Goals

1. **Working End-to-End Pipeline:** Real-time object detection → scene captioning → audio output
2. **Web Interface:** Accessible Next.js frontend with live camera feed
3. **Performance Benchmarks:** Comprehensive evaluation of latency, accuracy, and quality metrics
4. **Documentation:** Well-documented codebase with usage examples

Success Criteria

- **Quantitative:** Latency < 500ms (target exceeded: 194.4ms), Frame rate ≥ 2 FPS (achieved: 5.14 FPS), Detection mAP ≥ 0.40, Caption BLEU ≥ 0.30
- **Qualitative:** Working web interface, clear descriptions, smooth audio output, accessible UI

Timeline

Week	Milestone
Week 1	Setup, data exploration, YOLO11 inference
Week 2	VLM integration, TTS setup, backend API
Week 3	Next.js UI, end-to-end testing, documentation