Appendix A: Depth Estimation Model Comparison & Selection

Objective

The goal of this evaluation was to identify a depth estimation model that provides a good balance of **accuracy**, **inference speed**, and **resource efficiency** for integration into the SafeWalk AI pipeline, which is designed to run on **mobile or low-resource devices**. The depth model must generate **dense, pixel-wise depth maps** from a single RGB image in real-time, ideally without the need for dedicated hardware (e.g., depth sensors or GPUs).

Candidates Considered

We evaluated several existing pretrained models for monocular depth estimation, including multiple versions of **MiDaS**, along with a few other widely used alternatives: ✦ selected

Model Name	Source	Size	Inference Speed	Accuracy	CPU Performance	Notes
MiDaS v3 (DPT Large)	Intel-ISL	Large (~450MB)	Slow (GPU recommended)	High	✗	Accurate but too heavy for mobile
MiDaS v2.1 (Small) ✦	Intel-ISL	Small (~40MB)	Fast	Good	✓✓✓	Best balance; selected model
MiDaS v2.1 (Hybrid)	Intel-ISL	Medium (~120MB)	Moderate	Slightly better than Small	✓	Slower, minor gain in accuracy
Monodepth 2	UC Berkeley	Moderate	Moderate	Decent	✓	Requires stereo-like assumptions
DPT-Hybrid (MiDaS v3)	Intel-ISL	Medium (~110MB)	Slow	Very High	✗	GPU required for

						smooth performance
LeReS (CVPR 2021)	HKUST	Large	Moderate	High	✗	Good for 3D scenes, but heavy
ZoeDepth (by Intel)	Recent	Moderate	High	High	✗	State-of-the-art but not yet stable on CPU
RealSense Depth Sensor	Intel HW	Hardware-based	Real-time	Ground-truth level	✓	Not suitable due to hardware dependency

---

Evaluation Criteria

We evaluated each model based on the following:

- **Model Size**: Suitability for embedding in mobile applications
- **Inference Time**: Frame rate on CPU-only systems (Raspberry Pi 4 / laptop without GPU)
- **Relative Accuracy**: Compared to known depth layouts using indoor/outdoor test scenes
- **Stability**: Consistency of outputs in real-world noisy environments
- **Ease of Integration**: Framework (PyTorch/ONNX), pre/post-processing complexity

---

Experimental Observations

1. MiDaS v3 (DPT Large / Hybrid)

- Provides some of the best depth maps among all tested models.
- However, **very high VRAM usage** and **slow inference** on CPU-only machines (over 1 second/frame).
- Required downscaling images to 256x256 to be usable, which reduced its effectiveness.

- Best suited for GPU inference or offline processing.

2. MiDaS v2.1 Small (midas_v21_small_256.pt)

- The **fastest model** tested, with stable performance on CPU (under 300ms/frame).
- In real-time scenarios, this model handled both **indoor** and **outdoor** scenes well.
- Reasonable accuracy and consistency, especially when focusing on **relative depth** (as our use case requires).
- Very easy to integrate via Torch Hub and outputs were compatible with our post-processing logic.
- **Selected as final model** for deployment.

3. Monodepth2

- Slower than MiDaS Small on CPU.
- Requires stereo-like or well-lit input and performs poorly with complex shadows.
- Produced inconsistent depth on indoor scenes with varied lighting.
- Integration was more complex (needed camera intrinsics or stereo baselines in some variants).

4. ZoeDepth / LeReS / DPT-Hybrid

- Produced beautiful outputs on GPU-based benchmarks.
- Failed to reach acceptable frame rates or stability on low-power CPU setups.
- Would require model quantization or optimization pipelines to work on mobile — not ideal for our timeline.

5. Intel RealSense Depth Camera

- Provides hardware-accurate depth maps.
- Requires USB 3.0, drivers, and additional sensors not feasible for mobile-first or wearable applications.
- Ideal in lab setups but **not practical** for real-world field deployment.

Why We Selected MiDaS v2.1 Small (midas_v21_small_256.pt)

After extensive experimentation, we concluded that **MiDaS v2.1 Small** provides the best trade-off between:

- **Speed (suitable for real-time inference on CPU)**
- **Compact size (~40MB)**
- **Good relative accuracy** (sufficient for identifying nearest obstacles)
- **Low power/resource consumption**
- **Ease of integration** with Torch Hub and OpenCV pipelines

While other models provided marginal gains in depth detail or segmentation smoothness, they were either too heavy, required GPU acceleration, or could not meet real-time constraints — all of which are essential in a mobile assistive navigation system for the visually impaired.

Final Integration Notes

- The model takes RGB images resized to 256x256 as input.
- Output is a single-channel depth map normalized and scaled for bounding box evaluation.
- We used the **central region of each object's bounding box** to estimate object-specific distance while minimizing edge noise.