**Proposal Report: Real Time Monocular Depth Estimation on Edge-AI**

**1. Project Title**

**Monocular Depth Estimation System for 3D Perception and Collision Mitigation on Edge AI Devices**

**2. Executive Summary**

This project aims to develop a low-cost, deployable system for Monocular Depth Estimation using a single RGB camera and Edge AI devices such as the Raspberry Pi 5. By leveraging neural networks, the solution estimates depth from monocular images and provides real-time alerts through visual and audio feedback. The system is targeted for deployment in Caterpillar vehicles to enhance 3D perception, object detection, and collision avoidance in cost-sensitive and sensor-constrained environments.

**3. Problem Statement**

Accurate 3D perception is vital for autonomous driving and collision mitigation in Caterpillar vehicles. Traditional depth sensors like LiDAR or stereo cameras are often expensive or unavailable on all vehicles. This project addresses the need for an alternative: leveraging monocular cameras to infer depth using AI-based estimation models. The challenge is to achieve this efficiently on low-cost edge hardware while maintaining real-time inference and high accuracy, evaluated by Absolute Relative Distance Error metrics.

**4. Objectives**

* Develop a monocular depth estimation model optimized for deployment on Edge AI devices.
* Integrate the model with real-time visual/audio alert systems for operator feedback.
* Demonstrate system feasibility using the Raspberry Pi 5 and other low-cost AI hardware platforms.
* Utilize open datasets like KITTI/NYUv2 for model training and validation.
* Ensure scalability and affordability across Caterpillar’s fleet.

**5. Technical Approach**

**5.1 Data Source**

* **KITTI Dataset**: Used for training and benchmarking the depth estimation model.  
  [KITTI Dataset Website](https://www.cvlibs.net/datasets/kitti/)

**5.2 Model Development**

* Implement neural network-based depth estimation (e.g., MiDaS, Monodepth2).
* Optimize model using quantization/pruning for lightweight deployment.
* Frameworks: TensorFlow Lite, PyTorch Mobile, or ONNX Runtime.

**5.3 Deployment Architecture**

* **Input**: RGB video stream from Pi Camera or USB webcam.
* **Inference**: Edge AI device runs model to estimate pixel-wise depth.
* **Output**: If object is detected within critical distance, system activates:
  + Visual alert (LED)
  + Audio alert (Speaker or Buzzer)
  + Optional display output for debugging or feedback

**6. Hardware Components**

**6.1 Edge AI Device**

**Raspberry Pi 5**

| **Specification** | **Details** |
| --- | --- |
| CPU | Broadcom BCM2712, quad-core Arm Cortex-A76 @ 2.4GHz |
| GPU | VideoCore VII, OpenGL ES 3.1, Vulkan 1.2 |
| RAM | 1GB – 8GB LPDDR4X |
| Storage | microSD + PCIe 2.0 x1 (NVMe SSD support) |
| Connectivity | Wi-Fi 6, Bluetooth 5.0, Gigabit Ethernet, USB 3.0 |
| Camera Interfaces | Dual four-lane MIPI ports |
| Video Output | Dual micro-HDMI (up to 4Kp60 HDR) |
| GPIO | 40-pin header |
| Power | 5V/5A USB-C with PD |
| Dimensions | 85mm x 56mm |

**Justification**:

* High CPU performance, improved I/O throughput, and wide support for AI libraries.
* Lower cost and power consumption compared to alternatives like NVIDIA Jetson Orin Nano.
* Large developer ecosystem and software support.

**6.2 Additional Peripherals**

| **Component** | **Model / Specs** | **Purpose** |
| --- | --- | --- |
| **Camera** | Raspberry Pi Camera V2 (8MP) / Logitech C920 | Image input for depth estimation |
| **Power Supply** | 5V / 5A USB-C | Stable power for continuous operation |
| **Audio Alert Device** | Buzzer / Mini Speaker | Sound-based operator alert |
| **Visual Alert Device** | LED indicators | Warning signals based on proximity |
| **microSD Card** | Class 10 / UHS-1 | OS, data, and model storage |
| **Cooling** | Heatsink + Fan | Prevents thermal throttling during inference |
| **Enclosure** | Raspberry Pi Case with ventilation | Protection and heat dissipation |
| **Jumper Wires & Breadboard** | Generic prototyping tools | Prototyping alert system circuitry |
| **Optional Display** | HDMI screen | For monitoring/debugging |
| **USB Hub** | Powered USB Hub | Support for multiple peripherals |
| **Tripod/Bracket** | For camera stabilization | Consistent capture angles for model input |
| **Optional Sensors** | LiDAR/Ultrasonic (future expansion) | Supplement vision-based estimation if needed |

**7. Comparative Analysis of Edge Devices**

| **Device** | **Pros** | **Cons** |
| --- | --- | --- |
| **Raspberry Pi 5** | Affordable, good performance, strong community support | Limited GPU compared to NVIDIA Jetson |
| **Jetson Orin Nano** | Better GPU (CUDA support), superior AI inference speeds | Higher cost, power consumption, larger footprint |
| **BeagleBoard AI** | TI AM5729 SoC, onboard TPU, open-source focus | Smaller community, limited support for some AI tools |
| **TI TDA4VM** | Automotive grade, optimized for real-time AI | Higher cost and development complexity |

**8. Expected Deliverables**

* Depth estimation model trained and validated on KITTI/NYUv2.
* Software stack deployable on Raspberry Pi 5 and similar platforms.
* Prototype with LED/audio alert system based on proximity thresholds.
* Benchmark results using standard evaluation metrics (Abs Rel Error).
* Final report with hardware schematics, software documentation, and deployment guide.

**9. Impact & Justification**

* **Cost Efficiency**: Eliminates need for stereo or LiDAR sensors on all vehicles.
* **Scalability**: Compatible with a wide range of edge devices.
* **Operator Safety**: Real-time alerts reduce chances of human error and collisions.
* **Adaptability**: Can be extended to multiple vehicle types and terrain scenarios.
* **Open-Source Alignment**: Leverages widely available datasets and AI frameworks.

**10. Budget Estimate (Approximate in INR)**

| **Item** | **Unit Cost (INR)** | **Quantity** | **Total (INR)** |
| --- | --- | --- | --- |
| Raspberry Pi 5 (8GB) | ₹7,470 | 1 | ₹7,470 |
| Raspberry Pi Camera V2 | ₹2,490 | 1 | ₹2,490 |
| Power Supply + Peripherals | ₹2,075 | 1 | ₹2,075 |
| microSD Card (128GB UHS-1) | ₹1,660 | 1 | ₹1,660 |
| Buzzer + LED Kit | ₹830 | 1 | ₹830 |
| Heatsink + Case + Fan | ₹1,245 | 1 | ₹1,245 |
| Misc. (Breadboard, USB Hub, etc.) | ₹2,075 | 1 | ₹2,075 |
| **Total** |  |  | **₹17,845** |

**11. Consideration of RTOS for Task Scheduling and System Management**

**Purpose of RTOS in Embedded AI Applications**

A **Real-Time Operating System (RTOS)** is designed to ensure that tasks execute within strict timing constraints. It provides deterministic scheduling, task prioritization, and real-time responsiveness — crucial for applications requiring millisecond-level precision and guaranteed response times.

**Assessment for This Project**

This monocular depth estimation system involves:

* Capturing RGB frames from a camera.
* Running a neural network for depth prediction.
* Triggering alerts (visual/audio) based on detected object distance.

These tasks are **computational** in nature and run **sequentially or with minimal concurrency**, especially on **Edge AI platforms** like Raspberry Pi 5.

**Why an RTOS Is *Not Required* for This Project**

| **Criteria** | **Raspberry Pi with Linux** | **RTOS (e.g., FreeRTOS)** |
| --- | --- | --- |
| **Operating System** | Full Linux OS (Debian-based) | Minimalist real-time kernel |
| **Task Requirements** | Inference-based, soft real-time | Hard real-time |
| **Task Criticality** | Moderate (human response scale) | High (e.g., motor control) |
| **Ease of Development** | High (Python/C++, libraries) | Moderate to low |
| **AI/ML Support** | Extensive (TensorFlow Lite, PyTorch) | Limited |
| **Peripheral Support** | Wide (USB, HDMI, camera, etc.) | Limited (requires more effort) |
| **Cost/Complexity** | Lower | Higher (more customization needed) |

**Conclusion:**

Using a **full Linux OS on Raspberry Pi** provides:

* Sufficient timing accuracy for vision-based object detection and alerting.
* Rich support for AI libraries, camera drivers, and peripheral interfacing.
* Simpler and faster development, debugging, and deployment cycle.

Therefore, **RTOS is not essential** for this application unless:

* The system scales to **multi-sensor fusion with hard timing constraints**.
* **Mission-critical or safety-certified** real-time behaviour is required (like in automotive-grade production systems).

If such requirements arise, RTOS or hybrid architectures (Linux + microcontroller offloading[Arduino]) can be explored.