Object Property Detection Report

## Objective:

Recognize **an object** and its physical properties like **material , surface friction , dimensions and approx. weight** for initial gripper setting.

## Current Methodology:

Source : [**Object-Property-Detection**](https://github.com/Kaivalya192/Object-Property-Detection)

1. Capture image and segment object inside image ([batch\_yoloseg.py](https://github.com/Kaivalya192/Object-Property-Detection/blob/main/batch_yoloseg.py) or [rs\_yoloseg\_capture.py](https://github.com/Kaivalya192/Object-Property-Detection/blob/main/rs_yoloseg_capture.py))
2. Two way of object property extraction:
3. Caption the object ([captioning.py](https://github.com/Kaivalya192/Object-Property-Detection/blob/main/captioning.py)) and extract property by LLM ([ollama\_mistral.py](https://github.com/Kaivalya192/Object-Property-Detection/blob/main/ollama_mistral.py)).
4. Directly pass object to LLaVA ([ollama\_llava.py](https://github.com/Kaivalya192/Object-Property-Detection/blob/main/ollama_llava.py))

## Future Requirements:

1. **Integrate Active Learning for Improved Segmentation.**
2. **Improve prompt for better and structured results.**
3. **Fine-tuning LLMs with Domain-specific Data:**
   * Fine-tune models like LLaVA and Mistral on a domain-specific dataset containing labelled images and corresponding object property annotations, ensuring the models better understand the nuances of different objects in the target environment.
4. **Model Evaluation and Feedback Loop:**
   * Regularly evaluate the model's performance on validation sets and create a feedback loop to update the training dataset with examples where the model's predictions were incorrect or suboptimal.
5. **Multi-modal Fusion**:
   * Integrate data from different sensors (e.g., depth camera, infrared) to improve the accuracy of physical property estimation, allowing for better gripper adjustments based on more data points.
6. **Automated Dataset Generation**:
   * Build a mechanism for generating synthetic data for various objects and materials, enhancing training data diversity to improve model robustness and performance.
7. **Optimize for Low-power Edge Devices**:
   * Adapt the entire pipeline for deployment on edge devices like NVIDIA Jetson or Coral TPU, enabling real-time processing in low-power environments and reducing cloud dependency.