EN.601.661: Computer Vision Project Pre-Proposal

Title: Multi-Task Perception Model for Autonomous Vehicles Simultaneous Object Detection, Semantic Segmentation, and Depth Estimation

Spring 2025

February 17th, 2025

Team Members

- **Member 1: Minh Lê**
  JHED ID: mle20
- **Member 2: Ritwik Rohan**
  JHED ID: rrohan2
- **Member 3: Yi Lu**
  JHED ID: ylu174
- **Member 4: Yuhan Yong**
  JHED ID: yyong2

Project Idea

Motivation

Autonomous driving technology has become more and more popular nowadays as one of the most important applications of computer vision. In reality, it needs to process a lot of information efficiently and accurately to ensure safe driving. As a consequence, we aim to experiment the Hydranet model from Telsa at first and evaluate its current performance. We aim to use Hydranet as an inspiration and extend some model functions, if possible.

More specifically, we hope to derive a multi-task model that can perform three tasks at the same time: semantic segmentation, depth estimation, and object detection. This means that the model will analyze the road, understand distances, and detect objects all at once, which can be quite helpful in practice.

Training Process

Instead of training three separate models, we will use a shared encoder to extract important features from images and then use three separate decoders, one for each task. In this way, each task has its own specialized decoder while still benefiting from shared image features, making the system more efficient. For object detection, we will test three different decoder architectures (YOLO, SSD, and PointPillars) to see the one performing best in this setup.

The backbone of our model is based on transfer learning, where we use a pre-trained MobileNetV2 encoder that has already learned useful visual features from large-scale datasets.

Input and Output

INPUT: The input to our model is a single monocular RGB image, which is first preprocessed by normalizing pixel values to ensure consistency across different lighting conditions and environments.

OUTPUT: The final output of the model consists of a segmentation map, a depth map, and bounding boxes for object detection, providing a complete scene understanding from a single image.

Architecture

ENCODER (Body): The MobileNetV2 encoder extracts multi-scale feature representations from the input image, allowing the model to efficiently learn meaningful patterns. By using a shared encoder, we reduce computational costs while ensuring that all tasks benefit from a common feature extraction process. This makes the model more efficient and helps improve overall performance across multiple tasks.

DECODERS (Heads): Instead of using a single decoder for all tasks, we will build three separate decoders, each specialized for a different perception task. The semantic segmentation decoder utilizes RefineNet to generate a pixel-wise class map, helping the vehicle understand lane markings, road boundaries, and obstacles.

The depth estimation decoder also uses RefineNet model to predict depth values for each pixel, allowing the vehicle to understand distances and spatial relationships.

Each decoder will process the shared feature maps differently to generate bounding boxes, class labels, and confidence scores for detected objects. For object detection, we will experiment with three different decoders YOLO, SSD, and PointPillars, to determine which approach works best in this multi-task learning setup. Each decoder will process the shared feature maps differently to generate bounding boxes, class labels, and confidence scores for detected objects.

Evaluation and Expectations

To measure how well our model performs, we plan to use mean Intersection over Union (mIoU) for segmentation, Root Mean Squared Error (RMSE) for depth estimation, and mean Average Precision (mAP) for object detection. By testing different object detection decoders, we expect to find the best balance between accuracy and speed.

Datasets

To tackle the proposed problem, we will use data from the KITTI Vision Benchmark Suite of the Karlsruhe Institute of Technology and Toyota Technical Institute as a starting point to add object detection to the model, considering that the foundation code uses KITTI for segmentation and depth estimation as their dataset. The KITTI Object Detection dataset, part of the larger KITTI dataset) contains 14,999 images with 9 different classes (i.e., car, pedestrian, cyclist, etc.). In addition, KITTI's small-scale dataset allows for faster run time when testing the decoder.

Once we successfully integrated object detection into the model, depending on time constraints, we hope improve the performance of object detection in the model with BDD100K, a larger and more diverse database by Berkeley Artificial Intelligent Research. The BDD100K dataset consists of 100,000 images and 12 different classes (i.e., person, car, bike, etc.).

Computational Resources

To train our multi-task autonomous driving model, which integrates semantic segmentation, depth estimation, and object detection, the minimum computational requirement is 16GB VRAM. This number is estimated based on a batch size of 1. The final batch size for better training results can be adjustable and we will seek more help from the professor.

Component	**Estimated GPU Memory Usage**
MobileNetV2 Encoder	∼1GB
Semantic Segmentation Decoder (RefineNet)	3–5GB
Depth Estimation Decoder (RefineNet)	3–5GB
Object Detection Decoder (YOLO/SSD/PointPillars)	4–6GB

Table 1: Estimated GPU Memory Requirements for Training

Example Algorithms

Our model follows a **backbone (encoder) - head (decoder) architecture** for multi-task learning. A shared **encoder** extracts features from input images, which are then processed by three different **decoder heads** for segmentation, depth estimation, and object detection.

Backbone - Encoder

i) MobileNetV2 A lightweight convolutional neural network (CNN) used for efficient feature extraction.

- Used as the shared encoder in our model to extract meaningful features from input images, benefiting all three tasks (segmentation, depth estimation, and object detection).

Reference: MobileNetV2: Inverted Residuals and Linear Bottlenecks

Heads - Decoder

1. **Semantic Segmentation Head**

   i) RefineNet A multi-task decoder with Contextual Refinement Pooling (CRP) blocks for segmentation and depth estimation.

   - Used as the backbone decoder for semantic segmentation and depth estimation, ensuring detailed spatial information is retained while processing shared encoder features.

   Reference: RefineNet: Multi-Path Refinement Networks for High-Resolution Semantic Segmentation

2. **Depth Estimation Head**

   i) RefineNet A multi-task decoder with Contextual Refinement Pooling (CRP) blocks for segmentation and depth estimation.

   - Used as the backbone decoder for semantic segmentation and depth estimation, ensuring detailed spatial information is retained while processing shared encoder features.

   Reference: RefineNet: Multi-Path Refinement Networks for High-Resolution Semantic Segmentation

3. **Object Detection Head** We compare three different object detection decoders to determine the most effective approach for our multi-task model.

   i) YOLO (You Only Look Once) A single-shot object detection algorithm that divides the image into a grid and predicts objects directly.

   - One of the object detection decoders we will experiment with to determine its effectiveness in our multi-task setup.

   Reference: YOLO: Unified, Real-Time Object Detection

   ii) SSD (Single Shot Detector) An anchor-based object detection model that detects objects at multiple scales.

   - Used as an alternative object detection decoder to compare its efficiency and accuracy against YOLO and PointPillars.

   Reference: SSD: Single Shot MultiBox Detector

iii) PointPillars A 3D object detection method optimized for LiDAR point cloud data.

- Explored as a potential object detection decoder to assess its performance in our model, particularly for 3D object detection tasks.

Reference: PointPillars: Fast Encoders for Object Detection from Point Clouds