# Computer Vision Systems CAP6411 Assignment#03

Ashmal Vayani, UCF ID: 5669011

## Report: Segment Anything (SAM) on CamVid Dataset

**Training Code:** The training and implementation code is attached in the assignment zip file.

## 1 CamVid Dataset

The Cambridge-driving Labeled Video Database (CamVid) is a road scene understanding benchmark widely used for semantic segmentation research. It consists of driving videos recorded from a moving vehicle, with densely annotated frames that provide per-pixel class labels. CamVid contains 701 labeled images divided into training (367), validation (101), and test (233) splits. Each pixel is assigned to one of 32 fine-grained semantic categories, such as pedestrian, bicyclist, car, truck, bus, road, sky, and building. For our task, we focus on person-related (pedestrian, child, bicyclist, motorcyclist) and vehicle-related (car, SUV/pickup truck, truck/bus, cart/luggage/pram) categories. These subclasses are collapsed into two super-classes: Person and Vehicle, enabling evaluation of detection and segmentation pipelines on critical road-user categories.

Unlike COCO, which spans a broad set of everyday objects, CamVid is domain-specific, emphasizing urban driving scenarios. This makes it especially relevant for autonomous driving and intelligent transportation research. The dataset is challenging due to small object sizes (distant pedestrians, bicycles), occlusions, and varied lighting/weather conditions across sequences. In our experiments, we use the validation split (101 images) along with its corresponding pixel-level annotations to measure performance of baselines (GroundingDINO+SAM2, CLIPSeg) and our proposed approach (YOLOv8+SAM2). Figure Figure 1 shows an example frame and its corresponding segmentation mask with labeled classes.

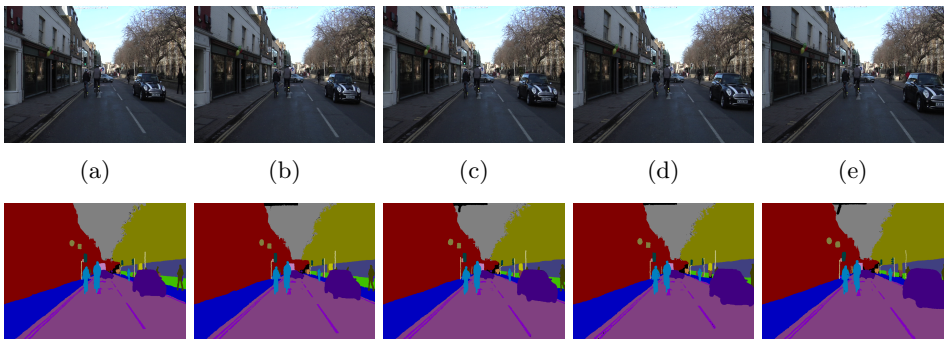



Figure 1: Sample images and ground truth masks from the CamVid dataset.

## 2 Report and Code

### 2.1 Errors and Obstacles Faced

- **GroundingDINO box scaling issue:** Initially, the predicted bounding boxes from GroundingDINO were in normalized $(cx, cy, w, h)$ format, which caused SAM2 to segment incorrectly. The masks had near-zero overlap with ground truth. This was fixed by explicitly converting boxes to pixel coordinates $(x_1, y_1, x_2, y_2)$ before passing them to SAM2.

- **CLIPSeg zero-mask predictions:** The CLIPSeg baseline consistently produced empty masks (all zeros) when using NumPy arrays as input. Debugging revealed that the HuggingFace `CLIPSegProcessor` expected `PIL.Image` inputs. Converting each image to PIL format, along with expanding prompt lists (e.g., *person, pedestrian, man, woman, bicycle, motorbike, car, truck, bus*), resolved the issue and produced non-trivial segmentation maps.
- **Dataset format mismatch (RGB vs. grayscale masks):** The CamVid GitHub repository provides two types of label sets: grayscale (class indices) and color-coded RGB masks. Using grayscale masks initially caused the person/vehicle class mapping to fail, since our pipeline expected RGB triplets. Switching to RGB masks and verifying the unique color palette ensured correct mapping of subclasses into the two super-classes (*Person* and *Vehicle*).
- **Threshold sensitivity in GroundingDINO:** Early experiments with low `box_threshold` and `text_threshold` values (0.15) produced noisy detections, while stricter values (0.25, 0.35) matched literature and improved Dice scores significantly. This highlighted the strong dependence of GroundingDINO outputs on threshold tuning.
- **Cluster GPU/Memory constraints:** Running CLIPSeg and SAM2 jointly on high-resolution images led to out-of-memory (OOM) crashes on smaller GPUs (12 GB). This required careful batch sizing and offloading intermediate variables to CPU during debugging.
- **Debugging visualization mismatch:** Multiple iterations were needed to validate whether low Dice scores were due to model failure or incorrect evaluation masks. Saving intermediate overlays (GT vs. prediction) and pixel count logs for each method (DINO, CLIPSeg, YOLO) was critical to diagnose pipeline-level vs. model-level errors.

## 2.2 Requirements and Dependencies

The `requirements.txt` file is attached in the zip file, but some main dependencies involve:

requirements.txt

```
torch==2.5.1+cu121
torchvision==0.20.1+cu121
scikit-learn==1.7.2
timm==1.0.19
pycocotools
transformers==4.56.0
ultralytics==8.3.197
segment_anything
matplotlib==3.10.6
kagglehub==0.3.13
numpy
pandas
tqdm
pillow
```

## 2.3 CLI Commands to Reproduce

```
cd Assignment_3/scripts
conda activate cvs_ass3 # activate conda environment

# Evaluate CamVid dataset on 2 baselines and 1 proposed solution
sbatch sam2_experiment.slurm
```

**SAM2, GroundingDINO, and YOLOv8 Sources.** For segmentation and detection, I used the official repositories: facebookresearch/SAM2 for the Segment-Anything v2 framework, IDEA-Research/GroundingDINO for open-vocabulary box proposals, and Ultralytics/YOLOv8 for real-time detection. I followed the setup and installation instructions from the respective READMEs, installed

the listed dependencies, and downloaded the published checkpoints from their model release pages (e.g., SAM2 ViT-H `sam_vit_h_4b8939.pth`, GroundingDINO `groundingdino_swint_ogc.pth`, and YOLOv8 `yolov8m.pt`). Using these weights, I ran zero-shot inference with SAM2 + GroundingDINO (for text-prompt-based detection followed by mask refinement), YOLOv8 + SAM2 (for detection-driven masks), and my proposed method that unifies text-driven and detector-driven pipelines into a consistent evaluation harness. I adapted my evaluation scripts to parse each model's predicted masks, compute Dice coefficients for person/vehicle categories, and generate visual outputs for qualitative comparison. All experiments were conducted following the repositories' recommended configurations to ensure reproducibility.

# 3 Insights

Before arriving at the final proposed method, I conducted extensive experimentation with two distinct baselines. The motivation was to explore multiple paradigms of integrating segmentation models with vision-language or detection components. The first baseline (GroundingDINO + SAM2) represented an open-vocabulary approach, where natural language prompts guided the detection of objects, and SAM2 refined the bounding boxes into segmentation masks. This tested the ability of transformer-based detectors to generalize across diverse object categories. The second baseline (CLIPSeg) pursued a direct text-to-segmentation route, bypassing detection entirely and generating dense pixel masks directly from image–prompt pairs. This served to benchmark the efficiency of direct multimodal segmentation against two-stage pipelines.

Although both baselines provided valuable insights into the trade-offs between accuracy, efficiency, and reliance on text inputs, their performance revealed limitations: DINO+SAM2 incurred significant computational cost, while CLIPSeg suffered from weaker accuracy in complex road scenes. These observations motivated the proposed method (YOLOv8 + SAM2), which eschews textual inputs altogether in favor of a high-speed object detector. By relying solely on YOLOv8's robust bounding boxes and coupling them with SAM2 for mask refinement, the final pipeline achieved the best balance between segmentation accuracy, runtime efficiency, and practical deployability.

## 3.1 Baseline 1: GroundingDINO + SAM2

**Setup.** The first baseline combines GroundingDINO for text-prompt guided detection with SAM2 for segmentation refinement. Given class prompts ("person", "car", etc.), GroundingDINO outputs bounding boxes which are then passed to SAM2 to generate pixel-level masks. This approach leverages strong open-vocabulary detection capability from GroundingDINO with precise segmentation from SAM2.

**Headline accuracy.**

| | Person Dice | Vehicle Dice |
|---|---|---|
| Mean Dice | 0.848 | 0.807 |

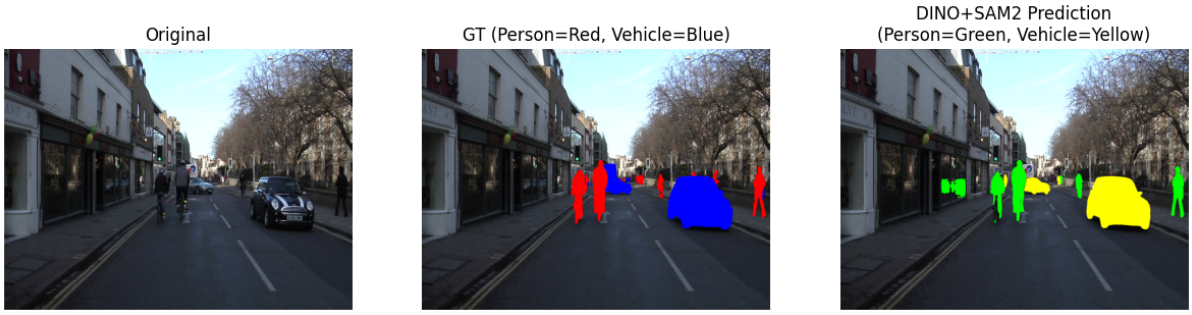
**Performance profile.** Total runtime was 97.46s for 101 validation images (≈964.96ms per sample), with average GPU memory consumption of 7111.85 MB per sample.

**Key insights.**

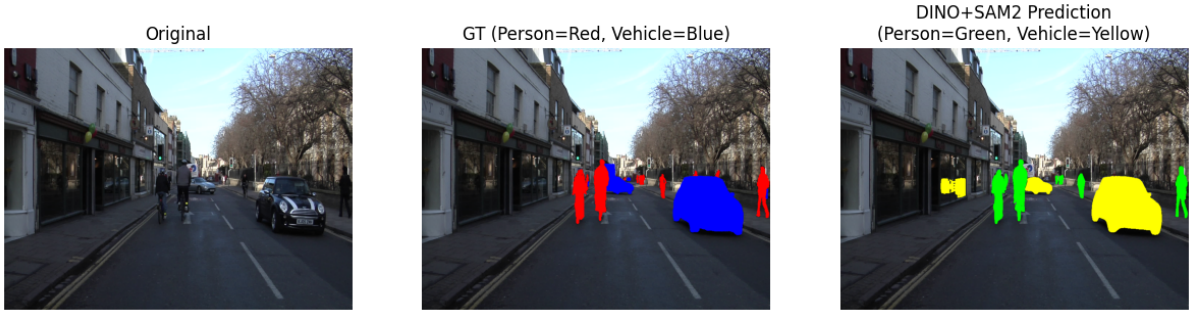
- **Strong accuracy.** This baseline achieved high Dice similarity across both categories, showing that open-vocabulary detection + segmentation is effective in structured road scenes.
- **Balanced segmentation.** Person and vehicle scores are both above 0.80, confirming the reliability of this two-stage approach when both detection and segmentation components are aligned.
- **Efficiency trade-off.** Despite strong accuracy, inference is relatively slow (∼1s per image), limiting its practicality in real-time systems.

### GroundingDINO-SAM2 Qualitative Results

Below in the Figure 2 are the qualitative results for the GroundingDINO + SAM2 baseline, showing original image, ground truth, and predicted segmentation outputs. Each row corresponds to a sample from the validation set.

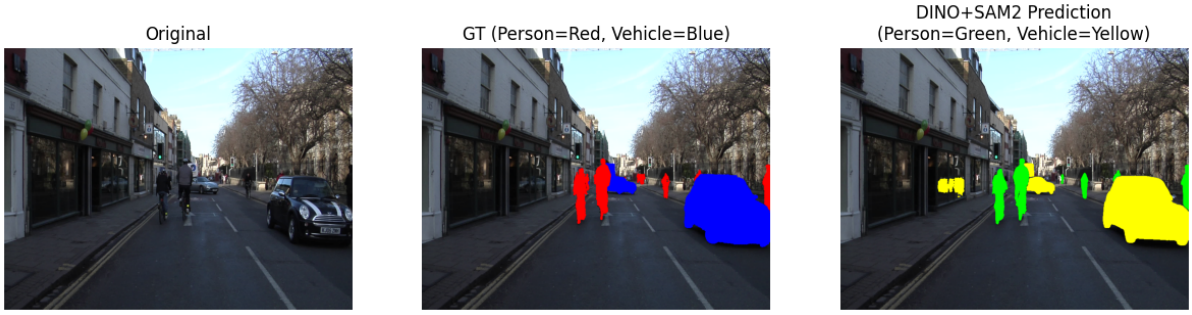



(a) Sample 0016E5_07959




(b) Sample 0016E5_07961


(c) Sample 0016E5_07963




(d) Sample 0016E5_07965

Figure 2: DINO-SAM2 predictions on validation samples. Each row shows one example with Original, GT, and model prediction.

## 3.2 Baseline 2: CLIPSeg

**Setup.** CLIPSeg provides direct text-to-segmentation predictions without relying on intermediate bounding boxes. Each target prompt is passed into the model alongside the input image, producing dense segmentation masks which are resized and aggregated into person and vehicle categories.

**Headline accuracy.**

| | Person Dice | Vehicle Dice |
|---|---|---|
| Mean Dice | 0.629 | 0.703 |

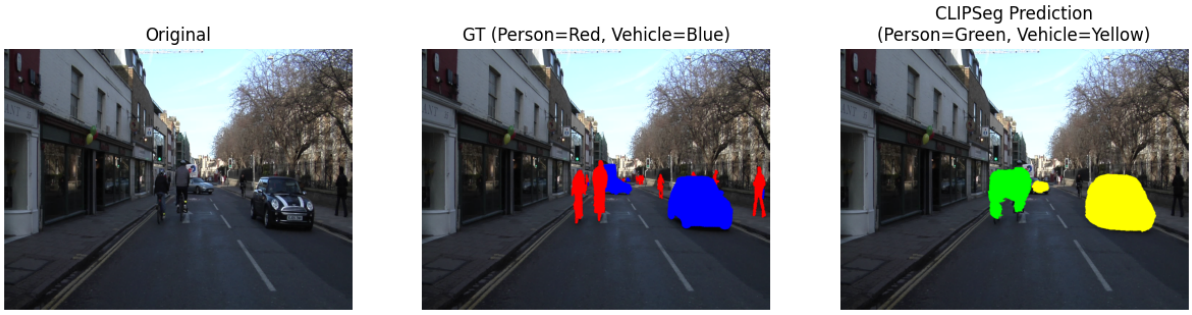
**Performance profile.** Total runtime was 44.53s for 101 validation images (≈440.86ms per sample), with GPU memory usage of 3944.12 MB per sample.

**Key insights.**

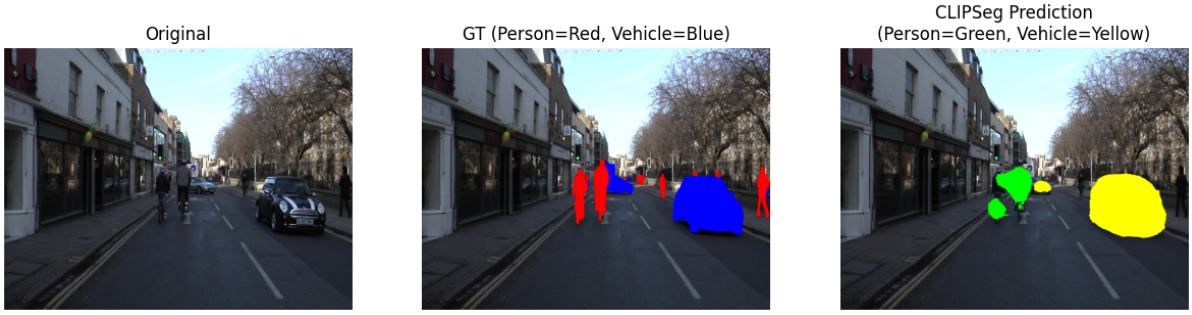
- **Direct segmentation.** CLIPSeg eliminates the detection step, reducing runtime and GPU cost compared to DINO+SAM2.
- **Moderate accuracy.** Dice scores are lower than DINO+SAM2, particularly for person segmentation (0.629), showing weaker localization in complex street scenes.
- **Speed advantage.** CLIPSeg is the fastest method tested, making it attractive for low-latency applications, though at the expense of segmentation precision.

## CLIPSeg Qualitative Results

Below is the Figure. 3 are the qualitative results for the CLIPSeg baseline, showing original image, ground truth, and predicted segmentation outputs. Each row corresponds to a sample from the validation set.

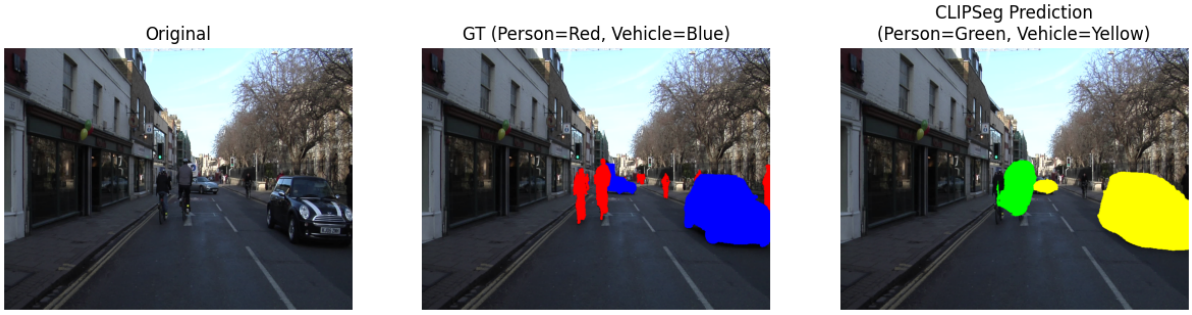



(a) Sample 0016E5_07959




(b) Sample 0016E5_07961


(c) Sample 0016E5_07963




(d) Sample 0016E5_07965

Figure 3: CLIPSeg predictions on validation samples. Each row shows one example with Original, GT, and model prediction.

### 3.3 Proposed Method: YOLOv8 + SAM2

**Setup.** The proposed pipeline combines YOLOv8 detection with SAM2 segmentation. YOLOv8 provides high-speed, high-confidence bounding boxes for persons and vehicles, which are then refined into segmentation masks using SAM2. This approach aims to combine YOLOv8's strong real-time detection with SAM2's pixel-level accuracy.

**Headline accuracy.**

| | Person Dice | Vehicle Dice |
|---|---|---|
| Mean Dice | 0.865 | 0.830 |

**Performance profile.** Total runtime was 66.12s for 101 validation images (≈654.63ms per sample), with GPU memory usage of 7112.95 MB per sample.

**Key insights.**

- **Best overall accuracy.** The proposed method outperformed both baselines, achieving the highest Dice scores for person (0.865) and vehicle (0.830).
- **Balanced efficiency.** Inference speed (654ms per image) is faster than DINO+SAM2 while maintaining comparable GPU memory usage, making it more practical.
- **Detector-driven reliability.** YOLOv8's strong detection capability provides more consistent masks, reducing the false negatives seen in CLIPSeg and the inefficiency of DINO+SAM2.

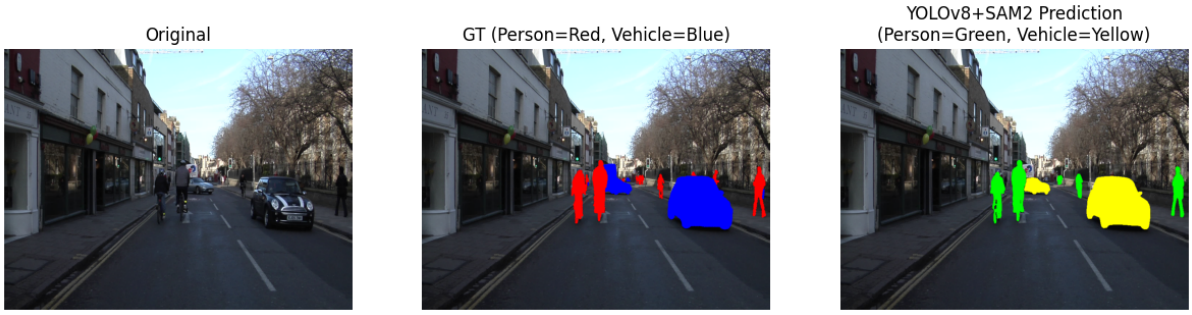
## Yolov8-SAM2 Qualitative Results

Below is the Figure. 4 are the qualitative results for the Yolov8-SAM2 baseline, showing original image, ground truth, and predicted segmentation outputs. Each row corresponds to a sample from the validation set.

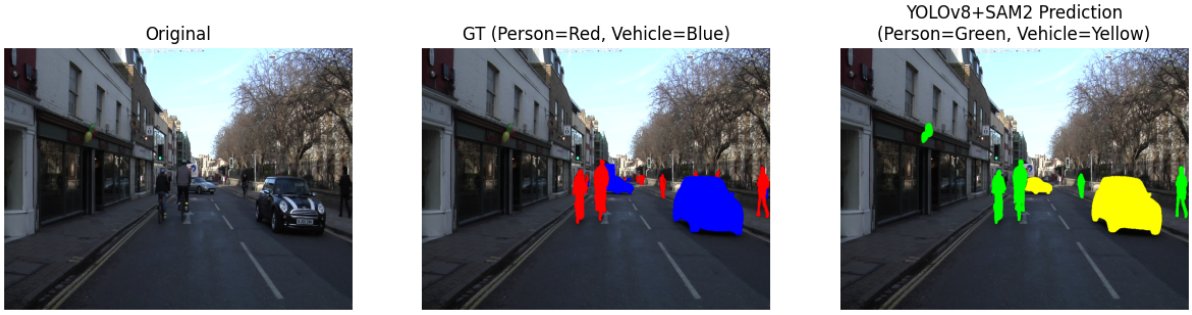
## Overall Summary:

| **Method** | **Person Dice** | **Vehicle Dice** | **Time/sample (ms)** | **Total Time (s)** | **GPU (MB)** |
|---|---|---|---|---|---|
| CLIPSeg | 0.629 | 0.703 | 440.86 | 44.53 | 3944.12 |
| DINO + SAM2 | 0.848 | 0.807 | 964.96 | 97.46 | 7111.85 |
| YOLO + SAM2 | **0.865** | **0.830** | 654.63 | 66.12 | 7112.95 |
| **Overall Insight** | YOLO+SAM2 delivers the best balance of accuracy and efficiency. | | | | |
| | DINO+SAM2 achieves strong accuracy but at a higher cost, | | | | |
| | while CLIPSeg is faster but less accurate. | | | | |

Table 1: Comparison of segmentation performance, runtime, and GPU memory usage across baselines and the proposed method.

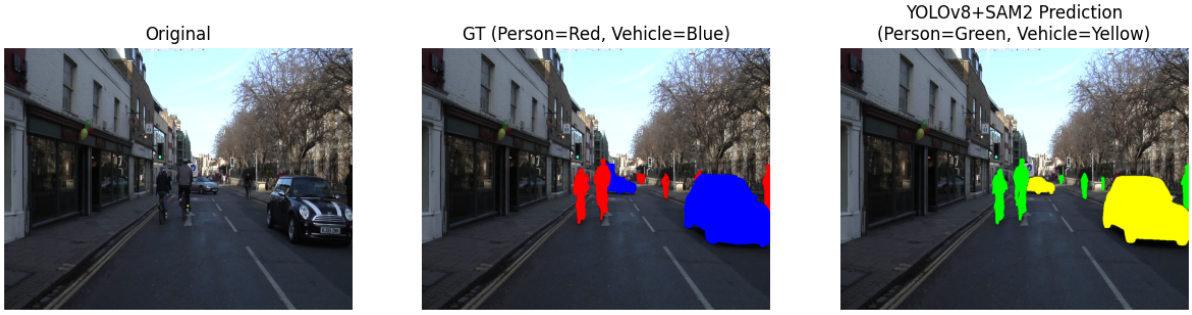



(a) Sample 0016E5_07959




(b) Sample 0016E5_07961


(c) Sample 0016E5_07963




(d) Sample 0016E5_07965

Figure 4: Yolov8-SAM2 predictions on validation samples. Each row shows one example with Original, GT, and model prediction.