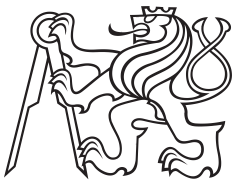
Bachelor Project



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

Rare event detection for autonomous driving

Kyrylo Herasymenko

Supervisor: Ing. Lukáš Neumann, Ph.D.
Study program: Software Engineering and Technology
May 2024

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Herasymenko Kyrylo** Personal ID number: **507335**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Computer Science**

Study program: **Software Engineering and Technology**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Rare event detection for autonomous driving

Bachelor’s thesis title in Czech:

Detekce vzácných událostí pro autonomní auta

Guidelines:

1. Familiarise yourself with large driving dataset provided by an industrial partner (Continental), analyse what data and sensor modalities are available in the dataset
2. Familiarise yourself with self-supervised vision transformer model such as DINOv2 [1], and with methods that use DINOv2 for Novel object discovery [2]
3. Build a pipeline to automatically look for certain parts of the large driving dataset using predefined criterion and verify the pipeline by looking at onboard sensor time series to find rare/unusual/surprising events in the measured data, for example sudden breaking or obstacle avoiding manoeuvre
4. Use the pipeline to process video recordings from a RGB camera through DiNOv2 [1] model
5. Use DINOv2 features generated in the previous step to look for rare/unknown objects, by looking for objects detected by DINOv2 but missed by standard object detectors such as Detectron [3,4], and analyse the results

Bibliography / sources:

1. Oquab, Maxime, et al. "DINOv2: Learning robust visual features without supervision." arXiv preprint arXiv:2304.07193 (2023)
2. Bharadwaj, Rohit, et al. "Enhancing Novel Object Detection via Cooperative Foundational Models." arXiv preprint arXiv:2311.12068 (2023)
3. HE, Kaiming et al., “Mask R-CNN”. Proceedings of the IEEE international conference on computer vision. 2017. p. 2961-2969
4. Wu et al., “Detectron 2”, online: https://github.com/facebookresearch/detectron2

Name and workplace of bachelor's thesis supervisor:

Ing. Lukáš Neumann, Ph.D. Visual Recognition Group FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **05.02.2024** Deadline for bachelor thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

Ing. Lukáš Neumann, Ph.D. Supervisor's signature	Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Acknowledgements

I would like to sincerely express my gratitude to my project supervisor, Ing. Lukáš Neumann, Ph.D., for his invaluable support throughout my work. I also thank Continental for providing the dataset.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

V Praze, 24. May 2024

Abstract

The main goal of this work is to compare standard object detectors with modern self-supervised object detectors in the context of vehicle driving. In the first part, two data subsets are created. The first subset consists of random video samples of driving, while the second subset is generated using a designed pipeline for detecting unusual events. These datasets are then processed using both a standard object detector and a self-supervised model. The results are compared and analyzed to determine whether objects undetected by the common detector can affect vehicle behavior.

Keywords: Anomaly detection algorithms, computer vision, self-supervised models

Supervisor: Ing. Lukáš Neumann, Ph.D.

Abstrakt

Hlavním cílem této studie je porovnat běžné detektory objektů s moderními samo-učícími se detektory objektů v kontextu řízení vozidel. Nejprve jsou vytvořene dvě datové sady: první sada obsahuje náhodné videozáznamy z jízdy, zatímco druhá sada je vytvořena pomocí navrženého postupu pro detekci neobvyklých událostí. Tyto datové sady jsou poté zpracovány pomocí standardního detektoru objektů a samo-učícího se modelu. Výsledky jsou porovnány a analyzovány, aby se zjistilo, zda objekty, které nejsou detekovány běžným detektorem, mohou ovlivnit chování vozidla.

Klíčová slova: Algoritmy detekce anomálií, počítačové vidění, samoučící se modely

Překlad názvu: Detekce vzácných událostí pro autonomní auta

Contents

Figures

Tables

Chapter 1

Introduction

The evolution of self-driving vehicles determined a new epoch in the automotive field, with a new approach to control the vehicle. It promises time efficient and safe drive. Nevertheless ensuring absolute safety still remains a challenge. Unexpected deviations from normal behavior, so-called anomalies can significantly reduce safety and can lead to road accidents causing vehicle damages and passenger injuries. In the field of autonomous vehicles, a key technology is the use of front-facing cameras. These cameras capture visual data from the environment, which is then processed by computer algorithms to inform the vehicle's behavior. However, pre-trained models used for object detection may not be able to identify all possible objects a vehicle might encounter. This research project aims to address this limitation by focusing on the detection of new and unexpected objects that could potentially impact the safe operation of autonomous vehicles.

1.1 Problem formulation

The main idea is to compare objects detected by common object detectors and self-supervised object detectors. Create two datasets with normal driving and unusual driving events. Process both datasets with a common object detector and self-supervised model. Select objects missed by the common object detector and determine which of these objects can affect vehicle behavior. For creating a dataset with unusual events, the next approach is used. A vehicle is equipped with telemetry recording sensors that take snapshots of the vehicle's state with a frequency of 17Hz. Given a dataset of records from

drives develop an automated system that identifies moments with deviations from the normal behavior of the driver. Download videos corresponding to the identified moments.

Chapter 2

Data

Continental has supplied a comprehensive dataset derived from cars driven on various roadways. Each vehicle in this study is equipped with data collectors, including a front-facing camera and a suite of sensors. These sensors are designed to collect a wide array of data points, such as vehicle velocity, acceleration, geographic position, turning angle, and other metrics that contribute to understanding the vehicle's performance and environmental interaction.
The vehicles were driven in various locations, but the majority of the data is concentrated in three main areas: Central Europe, North America, and Southern Asia. Specifically, the main countries where data were collected are Germany, the United States of America, Japan, and Italy.

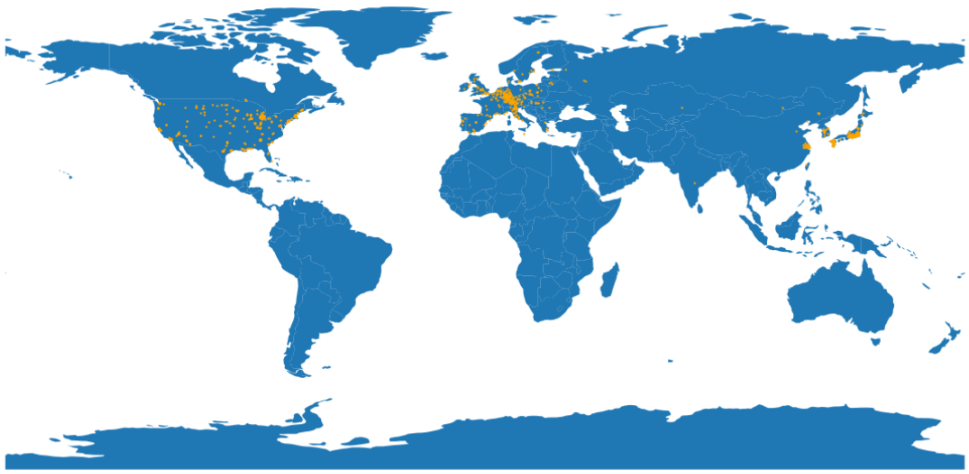


Figure 2.1: Drive locations

For a visual representation of the data, a map has been created where each drive is indicated by a point. This map, illustrated in Figure 2.1, offers a clear and immediate understanding of the distribution and density of the recorded drives across the mentioned regions.

2.1 Raw data organization

In the server-side storage, data is systematically arranged within directories. Each individual record is housed within its distinct folder, where the name of the folder serves as a unique identifier for the corresponding drive. Within these folders, these files are stored:

- vdy_synchronized.json
- device_urls.json
- cam_pose_calibration.json
- cam_intrinsic_synchronized.json
- drive identificator.mp4

Data were collected from 2,941 drives, totaling approximately 993 hours of driving. The total data size amounts to approximately 1.5 terabytes. Owing to constraints in available disk space on the work laptop, a reduction in data size was implemented during the download process. Specifically, only the JSON data files with drive records, denoted as "vdy_synchronized.json," were downloaded and subsequently renamed according to with their respective identifiers. Consequently, the resultant training dataset occupies 97 gigabytes of storage space. Overall information about data is Records about each drive

Drives	2941
Overall length (hours)	993
Overall space	1.5 TB
JSON files space	97 GB

Table 2.1: Data information

are stored in separate JSON files identified by filename. These files contain a collection of data points collected with a frequency of 17 Hz. Each data point is labeled with the following information:

- Vehicle velocity (m/s).
- Vehicle acceleration (m/s^2).
- Angle between front wheels and the direction of travel of a vehicle.
- Vehicle latitude (radians).
- Vehicle longitude (radians).
- gps_latitude - Vehicle latitude (radians) used only if gps_gen_pos_fix_latitude is unavailable.
- gps_longitude - Vehicle longitude (radians) used only if gps_gen_pos_fix_longitude is unavailable.

2.2 Subset sample

Due to limited computing capabilities, processing all 993 hours would take too long, so new smaller data subsets are created. First - with normal driving videos. The second is with strange events. To get objects from videos with normal driving, a new dataset called the Normal dataset is created by sampling random 350 sub-videos from the original dataset. Each subvideo has a length of 20 seconds, so the total amount of frames in this dataset is 119000. This dataset will be used in 5.

Another dataset, Driver Intervention dataset will be created in Chapter 4, section 4.3.5. This dataset will contain videos with unusual vehicle behavior.

Chapter 3

Related work

This chapter describes the existing solutions for object detection models.

3.1 Object detection

3.1.1 Detectron2

As a reference example of the existing object detector model, the Detectron2 [WKM+19] model by Facebook AI research is taken. This research project adopts Detectron2, a state-of-the-art object detection and segmentation library developed by Facebook AI Research (FAIR), as a reference model. As the Detectron2 GitHub repository states, "Detectron2 offers a suite of well-established algorithms, including Faster R-CNN [RHGS16], Mask R-CNN [HGDG17], and RetinaNet [LGG+18]. These algorithms have demonstrated high accuracy in various object detection tasks." The Detectron2 model offers image tagging, object localization, and object segmentation functions. The Detectron2 model includes implementations of models such as Mask R-CNN, RetinaNet, Faster R-CNN, RPN [RHGS16], Fast R-CNN [Gir15], R-FCN [DLHS23]. In this project, Mask R-CNN implementation is used. Mask R-CNN stands for Mask region-based convolutional neural network. It is an extension of Faster R-CNN adding an object mask to the output. It processes an image in two stages. The first, called RPN (Region Proposal Network) proposes bounding boxes, in which objects can be located. In the

next stage, all the proposed boxes are processed with RoIPool, and output features are used to perform classification and bounding box regression. In parallel, the second stage includes predicting a binary mask for each region of interest defined by the bounding box. The mask branch outputs K masks of size $m \times m$. Then, a per-pixel sigmoid is applied, and L_{mask} is defined as the average cross-entropy loss. For each region of interest, only L_{mask} associated with k-th class mask is considered. This allows to generate mask without competition among classes. The mask is predicted using a fully convolutional network [LSD15]. This method requires RoI features to be well aligned to preserve pixel-to-pixel correspondence. For this, mask R-CNN uses the RoIAlign [HGDG17] layer, which leads to large improvements. As a backbone, Mask R-CNN uses either FPN [LDG+17] or ResNet [HZRS16] networks.

This project uses the Detectron2 model implementing mask R-CNN with ResNet-50 and FPN backbone trained on the MS COCO dataset.
MS COCO [LMB+15] is an object-detection, key-point detection, segmentation, and key-point detection dataset. COCO stands for Microsoft Common Objects in Context and contains 328K images with per-instance labeled 91 object types.

3.1.2 YOLOv3

You Only Look Once version 3 (YOLOv3) [RF18] is a real-time object detection model known for its time-efficient image inference. YOLOv3 operates by predicting bounding boxes where potential objects can be located using a single forward pass through the network.

First, YOLOv3 predicts bounding boxes and their associated objectness scores. For this purpose, it uses dimension clusters as anchor boxes. The network predicts four parameters for each bounding box: t_x, t_y, t_w, t_h, which represent the coordinates and dimensions of the bounding box relative to the anchor box. The objectness score for

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 3.1: Darknet-53 network [RF18]

each bounding box indicates the likelihood that the box contains an object. This score is set to 1 if the box overlaps a ground truth object more than any other box. If the box overlaps a ground truth object by more than a certain threshold but not the most, the prediction is ignored.

Each bounding box then predicts the classes it might contain using multi-label classification, allowing for the possibility of multiple classes being associated with a single box. For feature extraction, YOLOv3 employs a new network called Darknet-53 shown at Figure 3.1, which has 53 convolutional layers and utilizes residual connections to enhance performance and accuracy.

3.1.3 RetinaNet

RetinaNet [LGG+18] is a state-of-the-art object detection model designed to achieve a balance between speed and accuracy. It is particularly known for its ability to handle the class imbalance problem in object detection through the use of the innovative Focal Loss function.

RetinaNet uses a single-stage architecture, meaning it predicts object locations and classifications in one pass through the network. This network is composed of a backbone network and two task-specific subnetworks. As a backbone, the FPN network is employed. Translation-invariant anchor boxes of areas 32^2 to 512^2 on pyramid levels P_3 to P_7 are used. Then, the classification subnet predicts the probability of presence of object for each of A anchors and for each of K classes. The classification subnet is a small FCN (Fully convolutional network) attached to each FPN level. Parameters of the subnet are shared between all FPN layers. In parallel, another small FCN is attached to detect the offset from the anchor box to a nearby object, if this exists.

3.2 Self-supervised models

3.2.1 DINOv2

DINOv2 [ODM+24] is a recent advancement in self-supervised learning developed by Meta AI department. DINOv2 itself is an effective model training method for variety of tasks in computer vision. In addition, DINOv2 authors released a family of high-performance pre-trained models. DINOv2 is applying the idea of using discriminative signals between images and group of images to learn features.

Dataset preparation

DINOv2 combines a large set of unlabeled data with images in a curated dataset to get a large training data. The pipeline of data processing is shown in Figure 3.2.

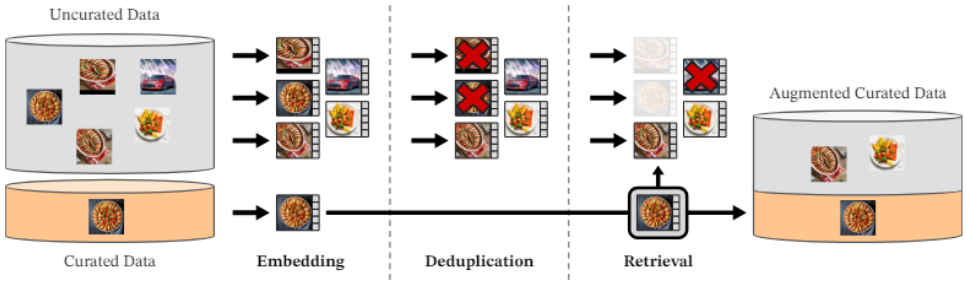


Figure 3.2: DINOv2 data preparation pipeline [ODM+24]

Pre-training

The features from images are learned using a combination of DINO and iBOT losses with the centering of SwAV [CMM+21]. Also, to spread features regularizer is added.

Implementation

The DINOv2 architecture relies on a pre-trained ViT model for feature extraction. Subsequently, features are processed with the model head. Pre-trained heads for depth estimation, semantic segmentation, and image classification are provided within DINOv2 GitHub.

Semantic segmentationis a computer vision task that involves classifying each pixel in an image into a predefined category. Unlike object detection, which focuses on identifying and localizing objects within bounding boxes, semantic segmentation provides a pixel-level understanding of the image, allowing for a more detailed analysis of the scene.

This project uses DINOv2 with backbone model ViT-large [DBK+21] model and head segmentator trained on ADE20K [BZT17] dataset.

ADE20K is a dataset used for semantic segmentation purposes. ADE20K dataset spans different annotations of images including the whole objects and object details and, in specific cases, parts of object details. The average scene in the dataset includes 29 annotated segments. The dataset consists of 20210, 2000, and 3000 images in training, validation, and testing sets.

3.2.2 RAM++

For image tagging with unseen objects, the RAM++ open-set model is used. RAM++ represents an advancement upon the foundational model RAM [HHZ+23] [ZHM+23] (Recognize Anything Model), which has established prominence in the domain of image tagging. The architecture of RAM includes three principal modules: an image encoder, an image-tag recognition decoder, and a subsequent text generation encoder-decoder. Specifically, the Swin-transformer [LLC+21] is employed as the image encoder, leveraging its efficiency in capturing spatial dependencies across image features. The text generation encoder-decoder comprises a 12-layer transformer architecture, employing comprehensive contextual understanding and generation of textual descriptions. Complementarily, a 2-layer transformer configuration serves as the tag decoder, tasked with accurately identifying and decoding image tags. This modular arrangement facilitates the nuanced processing of visual data, culminating in the proficient recognition and interpretation of images within the RAM++ framework. The RAM model is pre-trained on open-source datasets widely used in the image processing field. Among these datasets, there are MS COCO [LMB+15], Visual Genome [Kri17], and Conceptual Captions [CSDS21]. The output from the model is a list of tags both in English and Chinese languages.

3.2.3 Grounding DINO

Grounding DINO [LZR+23] Grounding DINO is an open-set object detector, which is obtained by combining transformer-based detector DINO with grounded pre-training. In general, Grounding DINO takes the pairs of (Image, Text) and detects objects from the Text on the Image. For each pair, Grounding DINO extracts vanilla text and image features using image and a text backbone. Then, features are processed with a feature enhancer module.

After obtaining cross-modality text and image features, a language-guided selection module is used to select cross-modality queries from image features. The output queries are used to predict object boxes for the corresponding tags from the text.

3.3 Tools used

This section lists all the libraries and tools used in the project.

Python
Python is a widely used programming language. Being considered slow among other programming languages, it still remains one of the most common programming languages for data analysis and processing thanks to large amount of easy-to-use tools and models written for Python.

NumPy
NumPy is a C-based Python module for data processing. In the project, NumPy is used for interpretation of driving records, images and image features as arrays.

OpenCV
OpenCV is an open-source image processing library. In the project, it is used for extracting frames from videos for their processing with models.

PyTorch
PyTorch is an open-source machine learning library developed primarily by Facebook's AI Research lab (FAIR). It provides a flexible and dynamic computational graph, allowing for efficient experimentation and model building. PyTorch is widely used for tasks such as deep learning, natural language processing, computer vision, and reinforcement learning.

Google olaboratory
Google Colaboratory is a web-based IDE for performing data analysis and data processing in Python. Providing a graphic runtime environment, it shows efficiency in processing images with PyTorch models. RAM++, GroundingDINO, and Detectron2 image processing was done in Google Colaboratory with T4 GPU.

Matplotlib

Matplotlib is a comprehensive plotting library for Python that produces high-quality visualizations for various types of data. It offers a wide range of customizable plots and can be integrated seamlessly with other libraries such as NumPy.

Chapter 4

Anomaly identification

4.1 Braking

Sudden brakings are considered unusual behavior on the road. In normal conditions, the driver stops slowly, so the self-driving car is not supposed to brake sharply. Hence sharp brakings can be taken as an anomaly in the self-driving scenario. This chapter aims to present two approaches to finding sudden braking scenes.

4.1.1 Acceleration minimum based approach

In the dataset, the braking action is presented by negative acceleration values, hence the lower is acceleration, the sharper is braking. The primary objective is to identify instances characterized by the lowest acceleration values. However, due to noise in the raw data, determining the precise acceleration at a specific moment poses a challenge. To address this, the Kalman filter is used, processing data sequences one by one and predicting subsequent values based on previous measurements. The Kalman filter returns the sequence of accelerations closely oscillating around their real values.

At the moment the objective is to find local minima and subsequently arrange them in ascending order. Because of oscillation after applying the

Kalman filter data is additionally flattened. In Figure 4.1 data before and after processing is shown. Raw data (orange), Kalman filter result (blue), additional flatted (red).

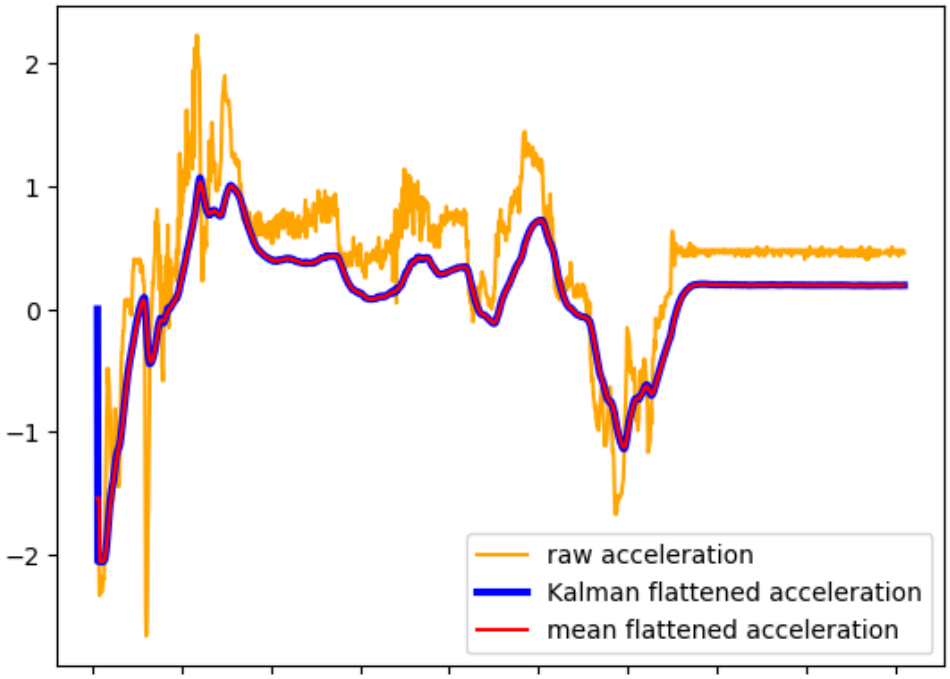


Figure 4.1: Processed acceleration

Next data is formatted into the two-dimensional array, with each row containing the drive identifier (JSON filename), the frame number (relative to the beginning of the record), and the acceleration value at that specific moment. Sorting this array by acceleration allows us to identify the moments with the most abrupt braking events. Upon the result of a single run, the algorithm gets the first 50 instances of sharp brakings and appends them to the main result list. After getting the main result list from all the files, the algorithm sorts by acceleration value and writes to the Excel file. The outcome is a table containing precise identification of every braking event. Next, it can be used for manual or automated video research. The implementation for this algorithm is stored in "accel_minimum_multithread.py". The next graph and image sequence 4.2 shows one of the moments identified by the algorithm.

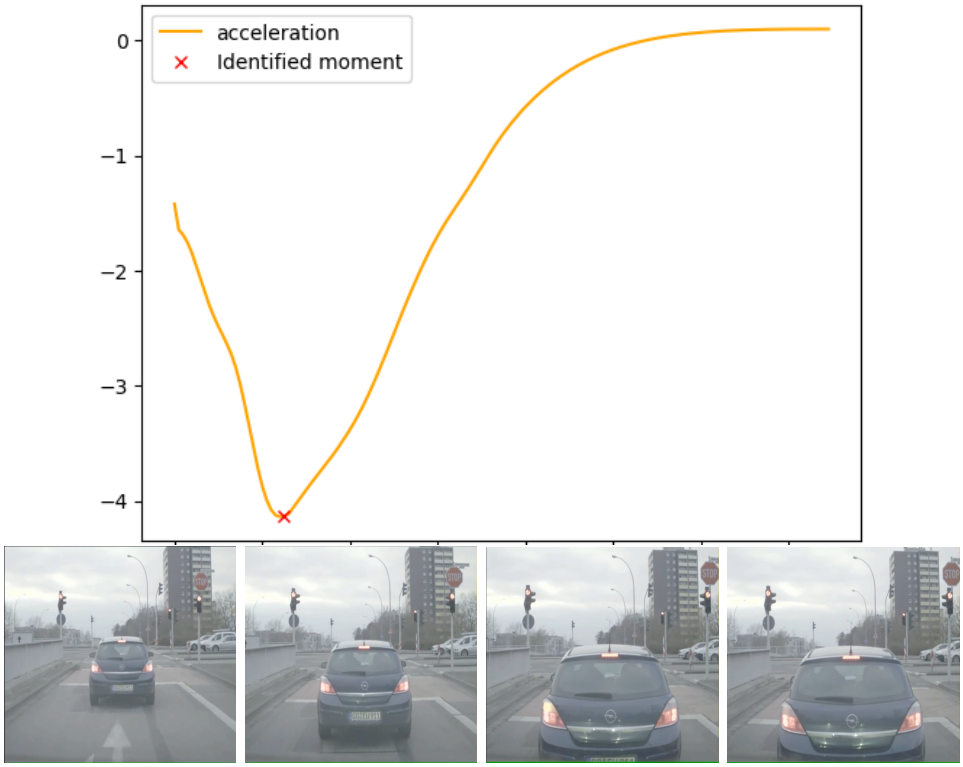


Figure 4.2: Braking sequence

4.1.2 Velocity pattern approach

Another approach to finding sudden brakings is finding a specific velocity pattern. If the vehicle is braking at t_i moment, v_{t_i} vehicle velocity is considerably lower than the mean velocity one second before. median$(\Sigma_{k=i-17}^{i} v_{t_k})$. A new variable difference rate is created. Next, it is written as d_i. To calculate d_i the next formula is used.

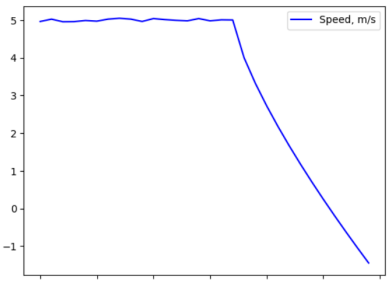


Figure 4.3: Velocity pattern

$d_i = v_{t_i} - \text{median}(\Sigma_{k=i-17}^{i} v_{t_k})$. This variable identifies how much velocity changed during the previous second. The smaller is d_i, the sharper is the brake. To cut off the moments, where the vehicle goes backward, the negative velocity condition is used. If $v_{t_i} <= 0$, then $d_i = 0$.

The image shows an example pattern the algorithm is looking for. After the calculation of d_i values, the moments with the smallest ones are chosen. Those

moments are then sorted ascending and pushed to the main result pool. The output of an algorithm is an Excel file with records about braking moments. Each record contains the unique drive identification (filename), moment (number of frames), and difference rate.

The current dataset is stored with sensor anomalies, where speed measurement is not recorded consistently, resulting in drops in velocity within a 1/17-second interval. The Kalman filter, used for data filtering, can not properly handle that sensor fails, so this approach leads to false positive outcomes. In Figure 4.4 an example of such a velocity fall is shown.

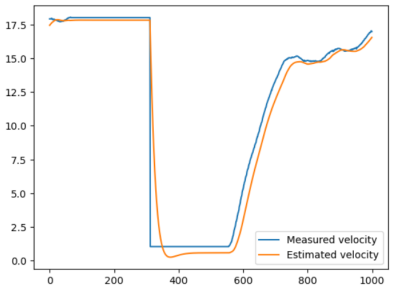


Figure 4.4: Sensor fail example

4.2 Sudden steering wheel rotation

4.2.1 Turn angle pattern

An algorithm described in this chapter uses the same approach as the algorithm described in section 4.1.2 - "Velocity pattern approach". The main goal is to find sudden changes in the steering angle. The algorithm processes steering angle data extracted from the input JSON file, calculating the difference d_i between the current steering angle and steering angle during the previous second. Mathematically that could be written as: $d_i = |\alpha_{t_i} - \text{median}(\Sigma_{k=i-17}^{i} \alpha_{t_k})|$
Here, α is the difference between the "zero" position of the steering wheel and its actual position. The next step is to label difference rates with their respective frame numbers.
During the next step, a velocity mask is calculated so only moments with a velocity higher than $8m/s$ are considered. Given the intent to identify local maximums in difference rates and considering the fact that the difference rate can not be negative, the difference rate of moments with a velocity of less than 8 is set to 0. After that, we reduce data so that only moments with local maximums in difference rates will remain. The result array is then sorted and appended to the main result array. That sequence is iteratively applied to all the files. At the end, the main result array is written to an Excel file. The script for this algorithm is stored in "turn_pattern_multithread.py"

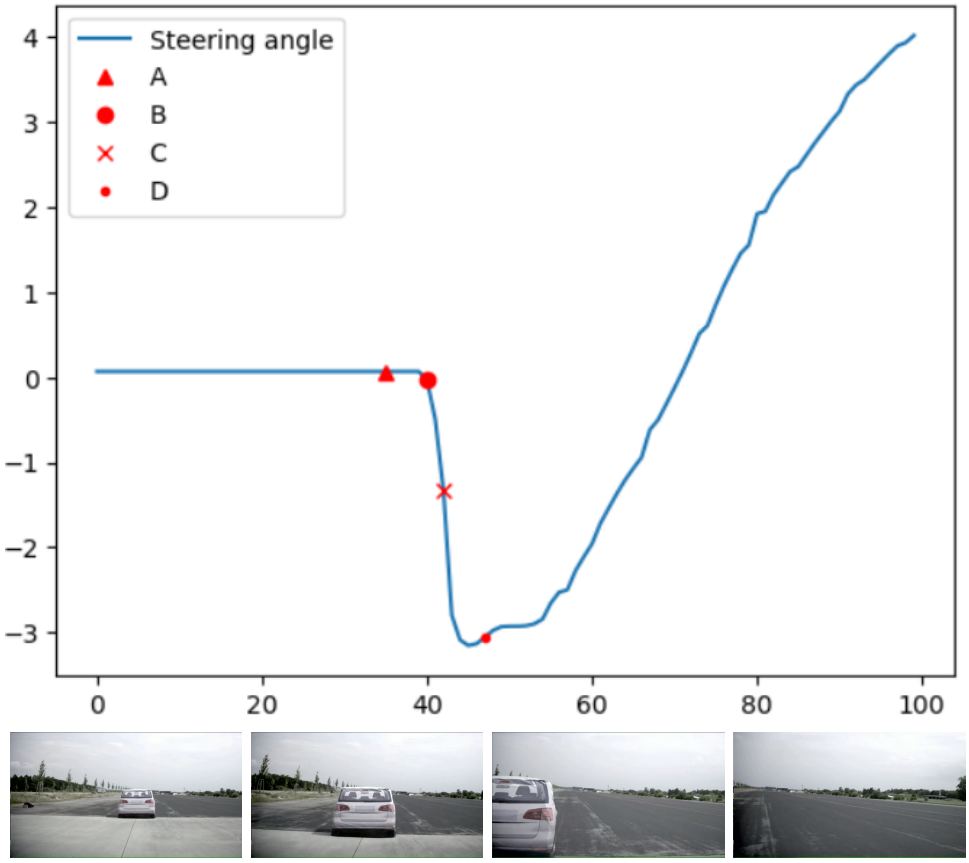


Figure 4.5: Turn sequence

4.3 Brake and turn combination

4.3.1 Algorithm

To measure overall danger at a specific moment, a new variable danger rate is included. This variable is written as r. Also, velocity difference rate v and turn difference rate t are used. To calculate it I use the formula

$$r_{t_i} = v_{t_i} + \alpha * t_{t_i}$$

Then local maximums of r are chosen and appended to the result array. Implementation of this algorithm is stored in file "turn_accel_multithread.py"

4.3.2 Alpha selection

Since α is not known, a comprehensive approach is taken by utilizing multiple α values and comparing the corresponding results. The methodology aims to retain the optimal α value, preventing the algorithm from overreacting to either sharp turns or sudden brakes. Given that v_{t_i} varies from 0 to approximately -5 and t_{t_i} has values between 0 and 5, α value should be around -1 for accurate computation of r_{t_i}. To evaluate the algorithm's performance across different *alpha* values, a set of [-2.5, -2, -1.5, -1, -0.8, -0.6, -0.4, -0.2] is used. Analysis of outcomes from an algorithm with different α values leads to the conclusion that $\alpha = -0.6$ is an optimal choice. Algorithms with α values exceeding -0.6 returned videos characterized by braking events, while algorithms with $\alpha < -0.6$ returned videos with low-speed scenarios, such as videos from parking, and small streets, where anomalies are unlikely to happen.

4.3.3 Dangerous places visualization

Following the computation of the danger rate for each frame within a recorded dataset, the identification of locations characterized by repetitive hazardous events and subsequent mapping thereof is undertaken. The main idea involves marking the boundaries of the map, partitioning it into cells, and calculating the overall danger rate of each cell. The function accepts parameters marking coordinates boundaries and the number of rows and columns for cell segmentation. Then it detects which drive record lies in the boundaries. A matrix sized according to number of rows and columns from the input is initialized to zeros. This matrix will identify the overall danger rate of the single square in the selected area. The position of the drive record is given by the first coordinates recorded. Then it processes all the files the way described in the Braking and turn combination chapter. After getting the result list, the algorithm filters it, retaining only those frames, where the difference rate surpasses a predetermined threshold. During the further step, the records detected are allocated to their cells by adding the danger rate to the corresponding cell in the matrix. The visual representation of these results is depicted in the accompanying images. Following the analysis of driving records, the two areas with the highest number of drives have been identified. These areas are Detroit and Tokyo.

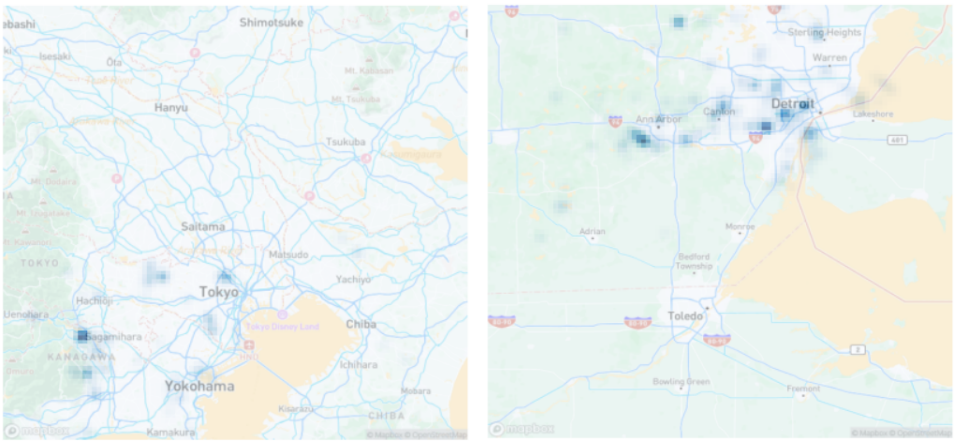


Figure 4.6: Visualization of dangerous places in Tokyo (left) and Detroit (right)

4.3.4 Video download

During the automated anomaly detection process, a complementary manual approach was applied. Following each execution of the algorithm, a new Excel file was created, containing drive identifiers, frame numbers within the drive, and an associated "anomaly rate." Subsequently, to check the existence of identified anomalies, a video downloader script was developed. The methodology involves downloading the video from the server, extracting only the pertinent moments (spanning 85 frames before and after the identified moment), and subsequently deleting the original video. As a result, the resultant folder exclusively contains video fragments representing the identified anomalies.

4.3.5 Result

At the end of this chapter, a set of video samples containing anomalies in driving are collected. These videos are forming a new dataset Driver Intervention dataset. This set includes 97 videos 10s each, so it consists of 16490 frames. These videos will be processed and compared to the processed Normal dataset 2.2 dataset in the subsequent steps to analyze objects that could potentially affect vehicle behavior.

Chapter 5

Object detector analysis

5.1 Object detector

5.1.1 Model description

To determine objects that are already identified by existing model,it is necessary to directly process videos using the model. That helps to determine, which objects are already detected, and which are ignored or misclassified. As a reference model, the Detectron2 [WKM+19] model is chosen.

5.1.2 Method description

The model offers detection and localization functions. To verify the detection (or potential non-detection) of objects, we will utilize the .pred_classes attribute within the model's image processing output. In order to detect objects, a procedural algorithm iterates through each video within a given dataset. For each video, a nested loop iterates through its frames sequentially. Detected objects are subsequently inserted into a map data structure, wherein the count of occurrences of each object is incremented. To reduce the probability of false positives, an object is appended to the result map only if it is detected in the consequent frames. For this, a data structure

queue is used. An array of objects detected is appended to the end of the queue. Then, to keep information from only two consequent frames, queue head is popped. After that, the queue returns the intersection of two arrays of objects detected, hence only objects that appear in both consequent frames are returned.

5.1.3 Objects detected

Videos processed are taken from two datasets: Normal dataset 2.2 and Driver Intervention dataset 4.3.5.

Normal dataset

Table 5.1 shows object classes detected during processing frames from the Normal dataset. Within objects that usually do not affect unusual behavior, like "car" and "truck", there are "traffic light" and "stop sign" which can cause sharp braking.

Name	Count	Name	Count	Name	Count
car	84974	fire hydrant	842	chair	229
truck	34125	potted plant	732	parking meter	182
person	19240	sports ball	705	boat	159
traffic light	17973	handbag	603	sheep	151
bus	7338	clock	597	sink	146
bench	3258	umbrella	521	dog	88
stop sign	1936	tv	440	cow	82
motorcycle	1621	frisbee	434	kite	76
bicycle	1317	airplane	318	bird	73
train	994	backpack	233	bottle	70

Table 5.1: Objects detected by the Detectron2 from Normal dataset

Driver Intervention dataset

The objects listed in Table 5.2 are extracted from Driver Intervention dataset frames. The object classes are similar, but the frequency of "stop sign" occurrences is higher - once at 28 frames in Driver Intervention dataset and once at 61 frames in Normal driving dataset.

Name	Count	Name	Count	Name	Count
car	8916	bus	489	bicycle	51
truck	2436	clock	168	sink	51
traffic light	2002	potted plant	98	chair	46
person	1512	motorcycle	94	train	41
stop sign	580	sports ball	83	fire hydrant	40

bench	558	kite	56	boat	35

Table 5.2: Objects detected by the Detectron2 from Driver Intervention dataset

5.2 Image features clustering approach

This section includes the results of processing the set of videos using DINOv2 model.

5.2.1 Designed pipeline

DINOv2 provides capabilities for both semantic segmentation and image labeling. The process involves a series of steps. Initially, semantic segmentation of video frames is conducted utilizing a pre-trained DINOv2 model. Simultaneously, feature vectors are extracted for each frame using the same DINOv2 model. Following this, connected components are identified to generate individual masks for each class. Next, connected components are filtered by their size. Too big components can symbolize noisy objects (like roads), whereas too small can be misdetected detail of something bigger, so they also add noise in the subsequent feature vectors array. Subsequently, for each

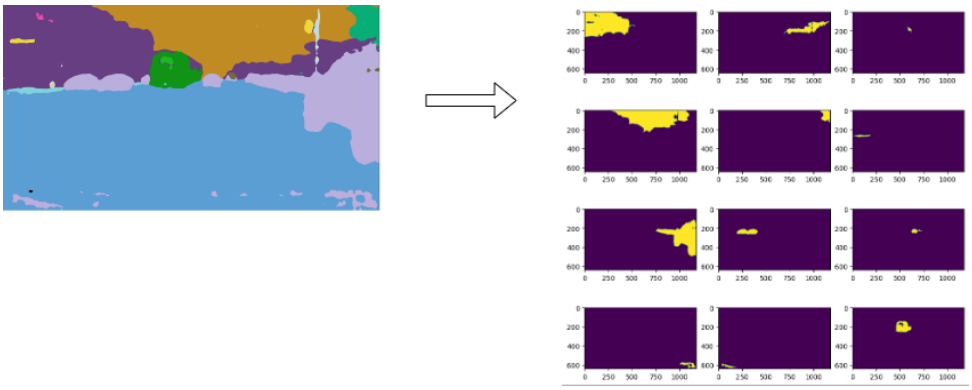


Figure 5.1: Connected components logics

connected component identified in the previous step, the mean feature vector is computed. The final step involves clustering the mean feature vectors, assuming that each vector corresponds to a distinct instance.

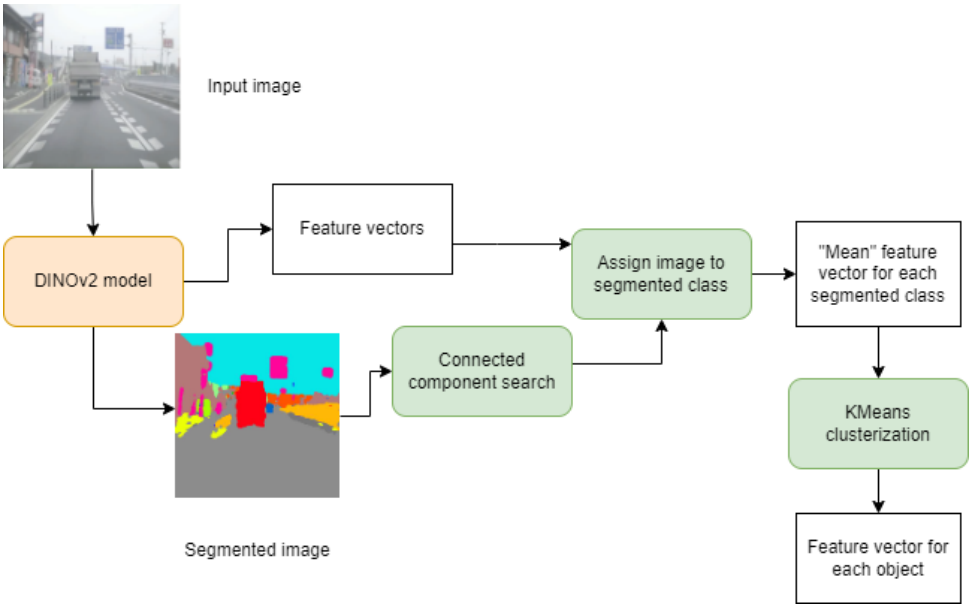


Figure 5.2: Image processing pipeline with DINOv2

5.2.2 Result

Step 3. of the designed pipeline assumes that semantic segmentation segments per instance, which would allow to correctly calculate and clusterize feature vectors for each instance. For example, with per-instance segmentation for the next input image, the output will be the image shows.

(a) : Input image

(b) : Actual semantic segmentation output

(c) : Expected per-instance output

Figure 5.3: DINOv2 output

After calculating the mean feature vector for each connected component, we tried to cluster the vectors with KMeans and determine the optimal number of clusters using the elbow method. However, the graph failed to preciously show the number of instance classes on the selected frame.

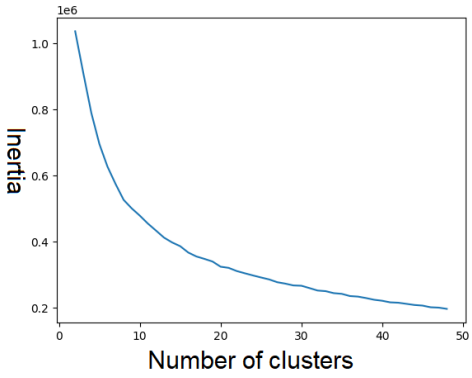


Figure 5.4: Clusterization error graph

5.3 Image labeling approach

This section focuses on getting novel objects using self-supervised RAM++ and Grounding DINO models.

5.3.1 Method description

To effectively extract novel objects from videos, the process can be compartmentalized into two sequential steps. Initially, the task involves identifying the objects depicted within the video frames, a task facilitated by the application of a "recognize anything" (RAM++) model. Subsequently, localization of the identified objects is undertaken, employing the GroundingDINO model to precisely determine the spatial coordinates of the selected objects within the video frames. This approach ensures a systematic and comprehensive extraction of novel objects from video data, combining object recognition with precise localization techniques for enhanced accuracy and effectiveness.

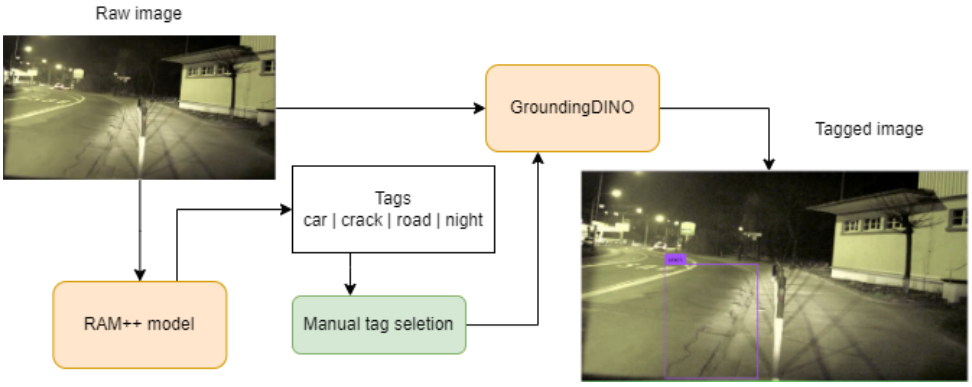


Figure 5.5: Processing images with RAM++ and GroundingDINO models

5.3.2 Excluded tags

Some tags are manually excluded due to misidentification or tag similarity. Objects, that were detected can be divided into several categories. Some of them are:

- "Snow" objects. Some frames are tagged by snow because of the low

quality of the recording camera.

- Misdetected objects
- Non-object tags, like night, red, evergreen
- Too common objects (road, cat, drive, etc.)
- Objects with the number of occurrences less than 50

5.3.3 Objects detected

Objects detected by the RAM++ model are shown in the tables below. The Normal dataset section reflects objects detected from 350 randomly chosen video samples. Driving with anomalies shows results from video samples in Chapter 4.

Normal dataset

After processing randomly sampled videos and excluding tags from the previous chapter, the next objects occurred. Due to the vast diversity of objects detected, only selected objects are shown. The full list of objects detected is shown in the table in appendix B.

Name	Count	Name	Count
street sign	16087	road sign	4698
tree	12173	fence	881
traffic light	8012	barrier	749
intersection	4845	zebra crossing	406
pole	4752	police car	149

Table 5.3: Object classes detected by RAM++ in Normal dataset

Driver Intervention dataset

Selected objects after processing video samples with driving anomalies are shown in the table below. All the objects are in the table in appendix B.

Name	Count	Name	Count
tree	4035	traffic light	741
street sign	3256	intersection	441

pole	1542	motorcycle	162
road sign	1098	stop sign	119

Table 5.4: Objects detected by RAM++ in Driver Intervention dataset

Objects detected in videos with normal driving and in videos with abnormal behavior are similar, but their frequency is different. The set Normal dataset has 119000 frames, and Driver Intervention dataset has 16490 frames. In Driver Intervention dataset, a tree occurs once in $16490/4035 = 4.08$ frames, whereas in Normal dataset its frequency is one tree per 9.7 frames. Street and road signs also occur more frequently in the Driver Intervention dataset dataset appearing once in 5 and 16.5 frames in videos with anomalies and once in 7 and 25 frames in the Normal dataset dataset. Other objects with a higher frequency are pole, stop light, fence. All these objects, especially road signs, can lead to braking or changes in driving direction and may be the reason why the video is included in the Driver Intervention dataset.

5.4 Object comparison

Lists of objects are manually processed and the next objects detected by RAM++ but not by Detectron2 are listed:

- Road sign
- Fence
- Barrier
- Cone
- Zebra crossing
- Tractor
- Excavator
- Ambulance

5.4.1 Undetected objects

Undetected objects refer to those that the current Detectron2 model fails to recognize entirely. Certain categories of objects have a notably greater impact on driver behavior compared to the classes predicted by the model.

Road sign
The road sign is an important class that can significantly affect the speed, vector of movement, or overall driver's behavior. However, only the "stop sign" can be detected by the Detectron2 trained on the MS COCO dataset.

(a) : RAM++ and Grounding DINO. **(b) :** Detectron 2

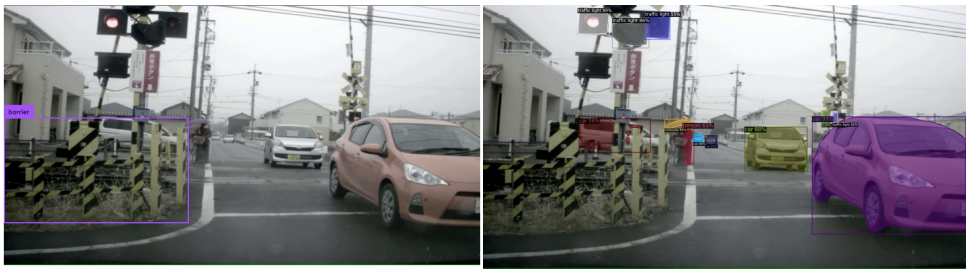
Figure 5.6: Road sign

Fence
The "fence" class also can not be detected by the Detectron 2 model. However, the presence of a fence signifies the absence of road infrastructure. Consequently, an undetected "fence" object holds the potential to cause road accidents.

(a) : RAM++ and Grounding DINO. **(b) :** Detectron 2

Figure 5.7: Fence

Barrier
The "Barrier" class is not included in the MS COCO dataset, it can not be recognized by Detectron2. The barrier function is similar to the fence function, but barrier can appear directly on the road, which has even larger influence on the driver's behavior.



(a) : RAM++ and Grounding DINO. **(b) :** Detectron 2

Figure 5.8: Barrier

Cone

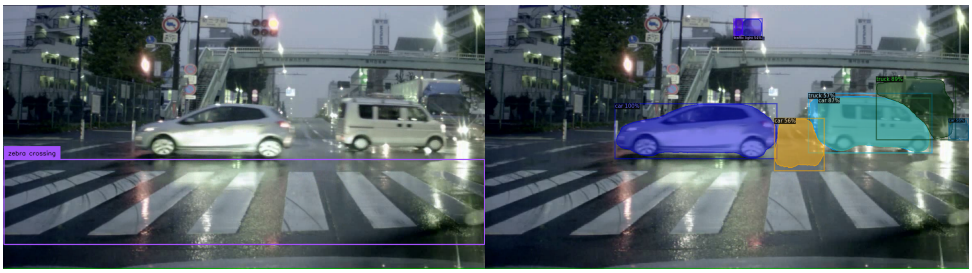
Cones placed on roads usually signal changes in road setup like lane divisions, the end of a lane, or ongoing road construction. They're brightly colored to catch drivers' attention, reminding them to be cautious and ready for any surprises ahead. However, since Detectron2 doesn't recognize cones, it can't process information about them accurately.

(a) : RAM++ and Grounding DINO. **(b) :** Detectron 2

Figure 5.9: Cone

Zebra crossing

A zebra crossing is a common feature on roads, designated for pedestrians to cross a road safely. To avoid collisions with pedestrians, drivers must be especially attentive and prepared to stop quickly if a pedestrian steps in front of their vehicle.



(a) : RAM++ and Grounding DINO. **(b) :** Detectron 2

Figure 5.10: Zebra crossing

5.4.2 Misclassified objects

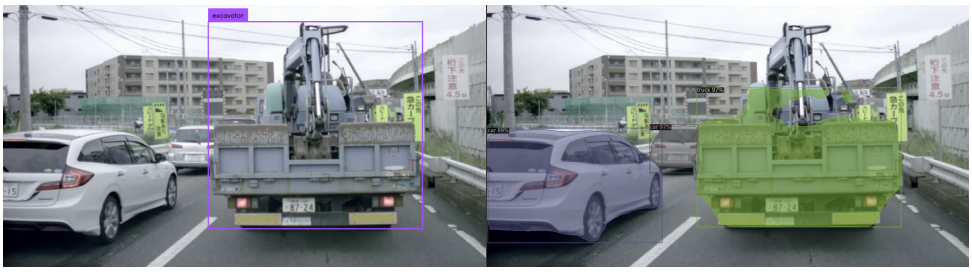
This section displays items that the Detectron2 model identifies, but they turn out to be different from the model output.

Tractor, excavator

The Detectron2 model mislabels tractors and excavators as "trucks," disregarding the distinct behavioral patterns exhibited by these vehicles. This difference arises from variations in road regulations governing the operation of tractors, excavators, and trucks, highlighting the need for precise classification in accordance with established vehicle classifications and corresponding road rules.

(a) : RAM++ and Grounding DINO. **(b) :** Detectron 2

Figure 5.11: Tractor

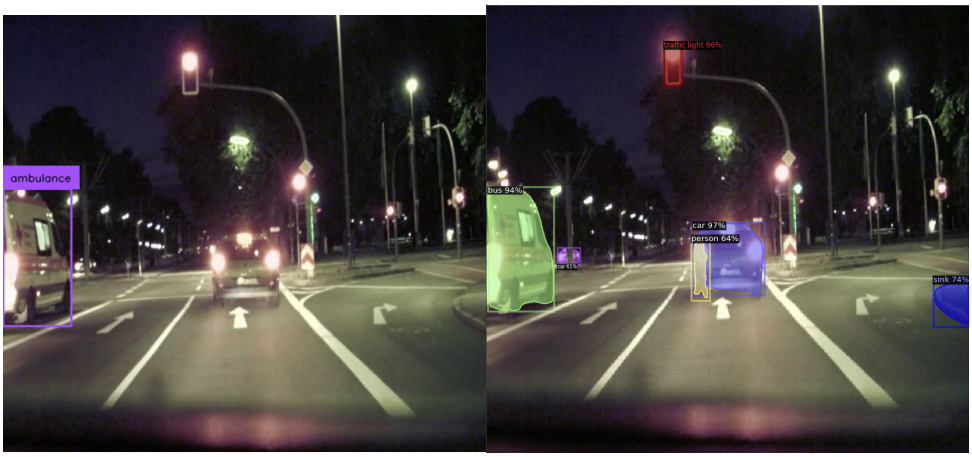


(a) : RAM++ and Grounding DINO. **(b) :** Detectron 2

Figure 5.12: Excavator

Ambulance

An ambulance is mistakenly identified as a "truck," which significantly affects driver behavior. If an ambulance is correctly identified, the driver might need to stop and give way to allow the ambulance to pass.



(a) : RAM++ and Grounding DINO. **(b) :** Detectron 2

Figure 5.13: Ambulance

Chapter 6

Conclusion

In this project, we designed and implemented a system to detect unusual events using data from car sensors. First, we identified anomalies by analyzing telemetry data, and then we processed the videos related to these events.

We used two different models to analyze the videos: a standard object detector called Detectron2 and a combination of the open-set RAM++ model and the self-supervised Grounding DINO model. We compared the results to identify which object classes were missed by the Detectron2.

In Section 5.4, we presented the objects that were missed by Detectron2 but detected by the RAM++ and Grounding DINO models and can potentially have an impact on the driver's behavior.

In summary, we resolved a set of object classes that can be added to existing object detectors. This helps in improving systems for advanced driver assistance and autonomous driving.

Appendix A

Bibliography

[BZT17] Xavier Puig Sanja Fidler Adela Barriuso Bolei Zhou, Hang Zhao and Antonio Torralba, *Scene parsing through ade20k dataset*, Computer Vision and Pattern Recognition (2017).

[CMM+21] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin, *Unsupervised learning of visual features by contrasting cluster assignments*, 2021.

[CSDS21] Soravit Changpinyo, Piyush Sharma, Nan Ding, and Radu Soricut, *Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts*, 2021.

[DBK+21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby, *An image is worth 16x16 words: Transformers for image recognition at scale*, 2021.

[DLHS23] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun, *R-fcn: Object detection via region-based fully convolutional networks*, 2023.

[Gir15] Ross Girshick, *Fast r-cnn*, 2015.

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick, *Mask r-cnn*, 2017 IEEE International Conference on Computer Vision (ICCV) (2017).

[HHZ+23] Xinyu Huang, Yi-Jie Huang, Youcai Zhang, Weiwei Tian, Rui Feng, Yuejie Zhang, Yanchun Xie, Yaqian Li, and Lei Zhang, *Open-set image tagging with multi-grained text supervision*, arXiv e-prints (2023), arXiv–2310.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, *Deep residual learning for image recognition*, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[Kri17] Zhu Y. Groth O. Krishna, R., *Visual genome: Connecting language and vision using crowdsourced dense image annotations*, International journal of computer vision (2017).

[LDG+17] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie, *Feature pyramid networks for object detection*, 2017.

[LGG+18] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár, *Focal loss for dense object detection*, 2018.

[LLC+21] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo, *Swin transformer: Hierarchical vision transformer using shifted windows*, 2021.

[LMB+15] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár, *Microsoft coco: Common objects in context*, 2015.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell, *Fully convolutional networks for semantic segmentation*, 2015.

[LZR+23] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al., *Grounding dino: Marrying dino with grounded pre-training for open-set object detection*, arXiv preprint arXiv:2303.05499 (2023).

[ODM+24] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski, *Dinov2: Learning robust visual features without supervision*, 2024.

[RF18] Joseph Redmon and Ali Farhadi, *Yolov3: An incremental improvement*, arXiv (2018).

[RHGS16] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, *Faster r-cnn: Towards real-time object detection with region proposal networks*, 2016.

[WKM+19] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick, *Detectron2*, `https://github.com/facebookresearch/detectron2`, 2019.

[ZHM+23] Youcai Zhang, Xinyu Huang, Jinyu Ma, Zhaoyang Li, Zhaochuan Luo, Yanchun Xie, Yuzhuo Qin, Tong Luo, Yaqian Li, Shilong Liu, et al., *Recognize anything: A strong image tagging model*, arXiv preprint arXiv:2306.03514 (2023).

Appendix B

Tables detailed

B.0.1 Objects detected by RAM++ in Normal dataset

Name	Count	Name	Count	Name	Count
street sign	16087	traffic jam	901	mountain	229
headlight	12314	fence	881	rail	220
tree	12173	pick up	856	park bench	220
overpass	9368	pine	835	bicycle	219
trailer truck	8562	floor	756	garbage truck	204
traffic light	8012	barrier	749	hill	202
truck	7774	path	725	ramp	189
park	7536	house	682	side	169
blanket	7240	taxi	672	police	167
city	6561	decker bus	658	umbrella	163
intersection	4845	stop sign	591	police car	149
pole	4752	motorbike	517	passenger train	147
road sign	4698	bus stop	517	store	145
traffic sign	4207	jeep	514	trailer	138
suv	4173	traffic	475	mound	127
city bus	4129	stand	473	minibus	125
windshield	3987	palm tree	431	scooter	123
van	3899	tow truck	419	biker	111
bus	3509	zebra crossing	406	can	101
stop light	2910	skateboarder	405	wood	95
sign	2681	sedan	403	excavator	87
bridge	2404	telegraph pole	400	garage	86
power line	2292	barricade	390	water tower	86

pillar	2070	trolley	382	crosswalk	82
tunnel	2061	fill	357	field	76
motorcycle	2015	cypress tree	356	cone	73
person	1788	crane	355	track	72
parking garage	1777	tank	355	bush	71
man	1696	cypress	330	stone building	70
parking	1489	dirt track	311	police van	69
ambulance	1472	car window	306	skateboard	67
street light	1439	neon light	281	officer	65
parking lot	1379	woman	274	skier	65
train track	1332	grove	267	parking sign	62
minivan	1294	view mirror	253	airliner	61
office building	1213	tour bus	250	plane	58
walk	1186	crack	249	dashboard	57
exit	1167	railroad	249	construction site	56
motorcyclist	927	desert road	248	bike lane	56
driveway	926	dirt road	244	recreational vehicle	53

Table B.1: Objects detected by RAM++ in normal drives

B.1 Objects detected by RAM++ in Driver Intervention dataset

Name	Count	Name	Count	Name	Count
tree	4035	bridge	268	city bus	115
street sign	3256	man	268	minivan	111
park	1888	car window	266	skateboarder	103
pole	1542	wood	263	pine	102
headlight	1522	van	249	bush	91
road sign	1098	skier	238	hill	86
traffic sign	900	dirt road	221	walk	86
sign	864	train track	211	jeep	84
blanket	857	floor	203	driveway	82
traffic light	741	office building	190	barrier	77
trailer truck	519	person	173	airport runway	76
stop light	464	rail	170	motorcyclist	73
fence	464	dirt track	167	passenger train	72
intersection	441	motorcycle	162	pick up	71
windshield	399	mountain	157	bus stop	71
truck	394	pillar	154	paling	63
parking lot	366	ramp	149	trolley	63

street light	344	bus	147	cage	63
house	333	stand	136	garage door	63
overpass	305	field	131	graveyard	58
suv	293	parking garage	129	container	57
path	279	stop sign	119	recreational vehicle	53

Table B.2: Objects detected by RAM++ in rides with anomalies