

Vision-Based Hazard Detection and Audio Alert System

Tyler Thach
Kevin Ruvalcaba
Bryan Zavala

Abstract

This paper analyses the effectiveness of combining real-time anomaly detection with LLM models. The combined model framework was based on the findings from other researchers which their work was assessed and expanded one with original additions. The model can identify various obstacles and give an audio description of what it sees. Depending on the severity of the obstacles, the audio description could be a general summary or a stern warning to stay alert of what is ahead. Through testing various environments, the model was quite accurate in detecting anomalies even in the densest of environments. However, many issues arose in the cost of running such a model as it took up a large amount of computational power to run. LLM's proved to be a great assistant with anomaly detection, but optimization is still needed for it to be effective in real-world scenarios.

1. Introduction & Problem Statement

The need for accessibility is ever increasing. With much of our infrastructure still needing major overhaul to make travel available for everyone, the field of Deep Learning may have solutions. Deep learning has advanced the capabilities for machines to obtain vision of their own. The power of computer vision has many usages but one that can truly pave the way for accessibility is anomaly detection.

With anomaly detection, the worry of running into obstacles lessens dramatically. It can be integrated into many applications such as Augmented Reality, robotics, or self-driving cars. It has been a technique that many have tried to harness to its full potential but runs into a few setbacks. In essence, it has tunnel vision and no ability to speak. To try to mitigate this, Large Language Models (LLMs) come in to assist.

The hope for LLMs to mitigate the pitfalls of object detection comes in the fact that these models are excellent at creating descriptions, describing data, and breaking it down. Many attempts have been made to effectively combine the two, but one research paper was noteworthy.

The paper, VisionGPT: LLM-Assisted Real Time Anomaly Detection for Safe Visual Navigation (et al) gives a solid basis as to how LLM-based object detection should operate.

In the paper, two models were used in combination: YOLO-World and ChatGPT. YOLO-World oversaw the handling of object detection which includes image processing, to which this data is given to ChatGPT to process the information. It then takes the information and does two things: decipher the information given to summarize to users and uses the information to create new prompts to feed into YOLO-World. It is a system that can rely on itself entirely [9].

However, this system left much to be desired in certain aspects. It lacked certain features that could make it worthwhile for a consumer and the efficiency of the system can be questioned due to how much energy ChatGPT can consume. As the idea of an LLM-Assisted Anomaly Detector is very compelling, further testing of the original framework model was done with unique data different from the original testing database to test the efficiency alongside the addition of new accessibility features.

2. Related Work

2.1. YOLO Challenges and Pitfalls

YOLO (You Only Look Once) models have been revolutionary when it comes to the field of object detection. It was able to take the area of object detection from a two-stage system to one stage. Newer models even implement a Zero-Shot model which requires no former training to be able to detect an object. However, YOLO models run into some challenges which is what makes the addition of LLMs a fruitful inclusion.

One of these challenges involves handling multi-scale training. Object detection relies on fixed resolutions. In the case of YOLOs, 448x448 image resolution is used while other models can utilize 512x512 or 640x640[9]. This is a decision made to make the training process efficient but can become an issue when handling input of

different resolutions. A common factor that can happen with live video.

Another challenge, which is one of the main ones for object detection, is the detection of small objects [7]. Given that the model is down sampling input, any small object that was in the given image can become obscured and even omitted. This issue is mitigated slightly through YOLOs' CNN model that accounted for these features from the initial convolutional layers. However, dense metropolitan areas can have a lot of clutter and make it harder to detect the finer details.

Given these challenges, the introduction of LLMs makes sense as they can offload some of the work the CNN model has to do. The LLM takes in the data given by the CNN, classifying and catching potential misses.

2.2. Fast Ego-Motion Prediction

To best make object detection worthwhile for consumer usage, it should be able to detect any potential harm. Current implementations work well with detecting stationary and objects in motion. However, unreliable motions may not be detected by these models. Usually, this means human motion. Motor Focus was developed to ease these issues.

Motor Focus is essentially an image-based framework used to detect human, and human-like motion (classified as Ego-Motion) through the usage of camera smoothness to filter out noise [3]. When tested, the method was able to capture motion from a moving vehicle while ignoring noise caused by real world factors such as shaky movement, and blurry quality [3].

Through methods such as Motor Focus, the quality of input given to an object detector may not be a limiting factor. Input will be clear enough that small details may be captured.

2.3. End-to-End Multimodal Model for Autonomous Driving

An example of LLM usage alongside image detection, End-to-End Multimodal Model for Autonomous Driving (EMMA) proves the successful implementation of such a hybrid approach. From the research team at Waymo, EMMA maximizes its learnt knowledge to represent non-sensor inputs as natural language text [7]. It takes in raw camera sensor data and converts them into outputs useful for making driving safer [7].

With Waymo being a prominent figure rising in the self-driving space, its usage of LLMs for their object detection shows its potential. It not only makes the detection model efficient as it is being fed clear and concise prompts from the LLM, but the user also gets concise information of what is occurring. Such information is crucial when having a computer operate such sensitive machinery. It comes to show the advancement of self-driving cars as early iterations relied on camera information only. Waymo has different data to go off on: Lidar, radar, and camera. All inputs that can be easily analyzed LLMs.

3. Methodology

In this project, we develop a real-time hazard detection and alert system based on a YOLO-World object detector. The major components that make up this model are object detection, scene segmentation, object prioritization, threat level estimation, and audio alert generation.

3.1. Object Detection

We utilize a YOLO-based model trained on a diverse set of objects to detect bounding boxes and class labels in each video frame. YOLOv8x-Worldv2 was the specific YOLO model we used. It is the largest v8 model, which means it is the most accurate but also the slowest. The model outputs predictions in xxyy format, class IDs, and associated confidence scores. For reliability, we apply a confidence threshold of 0.6, filtering out low-confidence detection. For each input frame I is an element of 3D real-valued tensors of shape $H \times W \times 3$ [9].

- **Predictions = {(xmin, ymin, xmax, ymax, class_id, confidence)}**
- **(xmin, ymin, xmax, ymax) defines box**

Our model supports zero-shot object detection, allowing it to recognize unseen objects based on textual descriptions without requiring training. This enables flexible adaptation to more obscure objects.

3.2. Frame Segmentation & Stabilization

To detect objects through location, we segment each frame into four key regions using a dynamic 'H-Split' structure. In the frame there is a left, right, front, and ground. We didn't want to be bound by static boundaries, so we incorporated a motion stabilizer that dynamically adjusts the horizontal splitting line based on the estimated motion center between consecutive

frames. This enables the system to maintain a better alignment with the real-world as the cameras' view shifts.



Figure 1: This is the H Splitter demonstrating how it prioritizes its areas.

Motion stabilization is achieved by computing an optical flow-based attention map between the current and previous frames, identifying the center of significant motion activity. Given two consecutive frames I_{t-1} and I_t , we estimate the motion center $(x_{\text{motion}}, y_{\text{motion}})$ and set the horizontal split at:

$$T_y = \text{clip}(y_{\text{motion}}, 0, H)$$

Where H is the image height. The vertical splits are placed at the static x -positions, L_x and R_x . Thus, any detection center (x_c, y_c) is assigned a region by:

$$\text{Region}(x_c, y_c) = \begin{cases} \text{Left}, & x_c < L_x \\ \text{Right}, & x_c > R_x \\ \text{Front}, & L_x \leq x_c \leq R_x \text{ \& } y_c < T_y \\ \text{Ground, otherwise} \end{cases}$$

This dynamic segmentation ensures that the ground zone remains closely aligned with the road surface even under camera movement, improving the reliability of hazard assessment [3].

3.3. Object Prioritization

After detecting objects, we prioritize them based on size and location. Objects that are larger, based on bounding box area relative to the frame, are assigned higher hazard importance. Objects located in the ground or front region are prioritized over those in the left and right regions. We computed this by:

$$\text{Area} = \frac{(x_{\text{max}} - x_{\text{min}})(y_{\text{max}} - y_{\text{min}})}{H \times W}$$

Objects with an area below the minimum threshold are ignored to reduce noise [3].



Figure 2: The H Splitter prioritizes the more hazardous objects and only recognizes objects that are close.

3.4. Danger Estimation & Post-Processing

For each frame there is a level score that is computed. The threat is determined by the presence of the object in hazardous regions, size of the detected objects, and the frequency and severity of alerts across multiple frames. There are only two types of threats, level 0 and level 1. 0 means safe and 1 means danger. When the threat level reaches 1, the system generates an alert [8].

After object detection and stabilization, a lightweight danger estimation module is applied to assess scene risks. A large language model is used to interpret object distributions and generate danger scores based on special arrangements and object types.

Specifically, categorized detections are fed into the LLM in template format, allowing it to recognize contextual risks, much like heavy obstruction in the path ahead, or objects appearing on the driving routes. The LLM output gives us the danger score and description of the hazard [8].

3.5. Audio Alert

When a hazardous object is detected, an audio warning is generated using text-to-speech (gTTS) [1]. The audio file includes pre-defined warnings such as, "Warning, hazard ahead. Please proceed with caution." The audio files are synchronized with the output video frame to provide real-time alerts to users.



Figure 3: Hold control and right click picture for audio alert.

4. Experiments

There were two different experiments that were conducted for this model. The first was to try and test how precise the model is. Second, we tried testing out different variables to see which was more accurate.

4.1. Experiment I

This experiment was conducted to investigate the effects of different confidence thresholds, filtering strategies, and frame skipping rates on the performance of the object detection and stabilization pipeline. The goal was to maximize precision and recall while maintaining real-time speed for specific scenarios.

The data set that was used was the COCO Validation set images. First we had to convert to YOLO format for ground-truth testing. We were trying to see the model's precision, recall, and F1-Score using ground-truth labels.

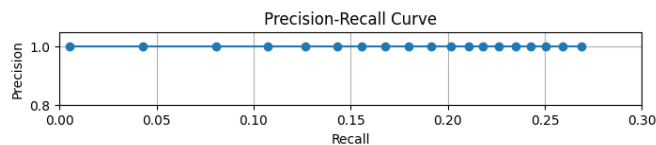


Figure 4: This was the plot point that was created to see if the change in threshold.

We tested out 19 different thresholds ranging from 0.05 to 0.95. It took about 70 minutes for it to fully compile.

4.1.1 Key Findings

The precision remained high across all the different thresholds 0.6 or above. By lowering the confidence threshold, recall improved significantly and recognizing smaller objects. Since these were images that were being tested the model had a hard time recognizing the smaller objects.

The best setting achieved was with a confidence threshold of 0.3 and an area filter of 0.02. Giving a result of 1.0 precision, 0.235 recall, F1 score of 0.380, and FPS of 23.5.

4.2. Experiment II

We conducted a series of experiments to investigate how different YOLO filtering thresholds, object size constraints, and frame skipping rates impact detection accuracy, system stability, and runtime efficiency. Specifically, we evaluated performance based on danger event detection, object counts, processing

speed, and overall runtime across both walking and driving scenarios.

Three video sequences were used, Walk1, Walk2, and Drive1. Walk1 features a calm sidewalk environment with minimal traffic or anomalies, providing a baseline for evaluation. Walk2 presents increased pedestrian and environmental activity, allowing for richer danger and anomaly detection. Drive1 captures a driving sequence through downtown San Diego, offering a more complex and dynamic setting for assessing model performance.

Exp	Conf	Area	Center	Frame Skip	Objects Detected	Danger	FPS	Time
W1_1	0.60	0.02	0.5h	6	73	0	14.35	1:22.5
W1_2	0.85	0.02	0.5h	6	18	0	15.04	1:09.7
W1_3	0.85	0.02	0.75h	6	73	2	15.04	1:21.4
W1_4	0.30	0.02	0.5h	6	73	1	14.80	1:19.9
W1_5	0.60	0.02	0.6h	5	73	1	14.80	1:19.9
W2_1	0.30	0.02	0.75h	6	131	12	14.38	1:33.1
W2_2	0.30	0.02	0.6h	6	131	12	14.72	1:27.8
W2_3	0.60	0.02	0.6h	6	131	12	14.72	1:27.8
W2_4	0.60	0.02	0.6h	6	131	5	14.32	1:21.4
W2_5	0.60	0.04	0.6h	5	131	19	15.07	1:21.2
D1_1	0.60	0.04	0.6h	6	199	10	14.70	4:53.4
D1_2	0.60	0.02	0.6h	6	464	27	14.70	6:06.8
D1_3	0.30	0.02	0.6h	6	464	27	14.49	5:32.2
D1_4	0.85	0.02	0.6h	6	253	10	14.06	4:59.3
D1_5	0.85	0.02	0.6h	5	464	162	12.15	6:04.0

Figure 5: The is the data chart for videos Walk1, Walk2, and Drive1.

4.2.1 Key Findings

Lowering the confidence threshold (CONF_THRESH) improved recall. After dropping the confidence threshold from 0.6 to 0.3, the number of detected objects remained high but did not significantly improve danger detection accuracy. Best performance came when CONF_THRESH = 0.6.

Minimizing the object size (MIN_AREA) made a difference. Setting MIN_AREA = 0.02 detected many more objects and more dangers compared to 0.04. A lower threshold is better for environments where small objects matter.

Frame skipping seemed to work out well. Skipping 6 frames provided a good trade-off between speed and responsiveness. FPS remained consistently at around 14.5, meaning that system was fast enough for near-real-time feedback.

There was a big difference in threat detection between the walking and driving scenarios. In walking scenarios, fewer danger events were detected compared to driving, which make sense because urban driving scenes are more dynamic and hazardous. Drive1 benefited the most from the fine-tuning.

The model without re-training generalized well to unseen street and walking scenes. Thus, demonstrating its effective zero-shot detection capabilities.

There is a tradeoff between speed and detection density. Slightly lowering thresholds increase detections but may increase false positives. Having, CONF_THRESH = 0.6 and MIN_AREA = 0.02, was the best balance between speed, object detection density, and real-world danger identification.

The best settings that were also the most accurate when it came to threat detection were CONF_THRESH = 0.6, MIN_AREA = 0.02, MAX_CENTER_Y = 0.6, and skipped frames at 6. These settings averaged 14.7 FPS. The most accurate videos when it came to threat detection were Walk1_3, Walk2_4, and Drive_4. These were tested by manually detecting the threats and comparing them.

5. Conclusion

The goal of this project was to develop and evaluate a stabilized object detection system designed to monitor hazards in walking and driving environments. Running experiments across different thresholds, object size, and stabilization settings, we found that a confidence threshold of 0.6 with a moderate center constraint produced the best overall performance, particularly in driving scenarios where hazard density was higher.

Motion stabilization enhanced detection reliability by adjusting to scene dynamics, while integrating a LLM provided real time scenario descriptions of dangerous events. Despite achieving higher precision, recall rates indicated some dangers were missed,

suggesting that the model could use some fine-tuning and specific training to improve its overall performance.

6. New Ideas & Improvements

When building and working on the model there was a lot that could have been really expanded on. Originally, there was going to be traffic light detection but things happened so there was a change in plans. That's a component that would really elevate this project into something super useful.

Building and training a YOLO model from scratch is something that seems really challenging but super rewarding. We did start the model, but the training would have taken weeks. So, building and training a YOLO model or LLM for this project would make the predictions much more accurate. The models could be trained for very specific tasks, such as crossing a busy street.

The audio file still is not perfect with some false negatives. It also calls out for danger too often. So fine tuning the model and improving the accuracy of the danger detection should be the next step to getting an extremely accurate model.

Another application that this project could be used is a very specific object detector. One that could pinpoint objects from very far and/or with very little information.

7. Team Contributions

As a team, we worked together to implement the model presented by the VisionGPT paper with our own contributions. Team member Tyler worked on the object detection aspect of this paper. He worked with the YOLO-World model to analyze the dataset given. He also tested the model via the H splitter method to compare results. Text-to-Speech was also tested by him. Many difficulties were had when compiling the model due to limited computational resources. It was also difficult to get certain aspects to get captured by the YOLO model.

Bryan worked on the LLM aspect. He worked with ChatGPT to have the appropriate prompts to send to the YOLO Model. He also worked on creating the output descriptions once the input was processed. It was a challenge to be able to separate from a general description to an anomaly warning.

Kevin worked on dataset preparation and project presentation and design. To differentiate from the original dataset, Kevin made the decision to incorporate data captured from local sites to test how effective the model was. Kevin also worked on the design of the PowerPoint presentation in terms of organization and data order, Introduction, Related Works, and Team contributions sections of this paper also written by Kevin.

References

- [1] A. Patil, A. Lule, V. Sathe, A. Borse, and P. Narkhede, "Object detection and conversion of text to speech for visually impaired," *Proceedings of the 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*, pp. 649–653, IEEE, 2021.
- [2] Diwan, T., Anirudh, G. & Tembhurne, J.V. Object detection using YOLO: challenges, architectural successors, datasets and applications. *Multimed Tools Appl* 82, 9243–9275 (2023). <https://doi.org/10.1007/s11042-022-13644-y>
- [3] H. Wang, J. Qin, X. Chen, A. Bastola, J. Suchanek, Z. Gong, and A. Razi. "Motor Focus: Fast Ego-Motion Prediction for Assistive Visual Navigation." *Proceedings of the 2024 IEEE 20th International Conference on Body Sensor Networks (BSN)*, pages 1–4, 2024.
- [4] H. Luo and P. Wei. "Binary classification with confidence difference." *arXiv preprint arXiv:2010.04947*, 2020.
- [5] J. R. Terven and D. M. Córdova-Esparza, "Object detection using YOLO: challenges, architectural successors, datasets and applications," *Multimedia Tools and Applications*, vol. 81, no. 8, pp. 9243–9275, 2022, doi: 10.1007/s11042-022-13644-y.
- [6] T. Diwan, G. Anirudh, and J. V. Tembhurne, "A comprehensive review of YOLO architectures in computer vision: from YOLOv1 to YOLOv8 and YOLO-NAS," *Multimedia Tools and Applications*, vol. 83, pp. 17521–17551, 2024, doi: 10.1007/s11042-024-17724-0.
- [7] Y. Cha, S. Lee, D. Yoon, S. Yoo, and M. Seo. "EMMA: End-to-End Multimodal Model for Autonomous Driving." *arXiv preprint arXiv:2401.04173*, 2024.
- [8] Y. Wu, Y. Liu, Y. Zhang, Y. Wang, F. Yang, C. Wang, and J. Jia. "VisionGPT: LLM-assisted real-time anomaly detection for safe visual navigation." *arXiv preprint arXiv:2401.01466*, 2024.
- [9] Z. Ma, Y. Zhang, F. Yang, H. Zhao, Z. Gao, Y. Wu, and J. Jia. "YOLO-World: Real-Time Open-Vocabulary Object Detection." *arXiv preprint arXiv:2304.00501*, 2023.