Available online at www.jpit.az

16 (2)
2025

SCOUT: A Solution to Automatic Inspection of the Outdoor Advertisement Banners

Jamaladdin Hasanov[a], Nizami Guliyev[b], Toghrul Babayev[c], Eljan Abdullazada[d], Vagif Karimli[e]

[a,b,c,d,e] School of IT and Engineering, ADA University, 1008, Ahmedbey Aghaoglu 61, Baku, Azerbaijan

[a]jhasanov@ada.edu.az, [b]nguliyev15981@ada.edu.az, [c]tbabayev14289@ada.edu.az, [d]eabdullazada14066@ada.edu.az, [e]vkarimli15756@ada.edu.az

[a] https://orcid.org/0000-0002-5437-7538

ARTICLE INFO

Keywords:

Object detection
Boundary detection
Banner recognition
Transfer learning
Llm
Marketing automation

ABSTRACT

This paper presents the design and implementation of SCOUT - Street-level Capture & Overlay for Urban Track-inspection - a hybrid AI system developed for the automated detection and geospatial mapping of outdoor advertisement banners in urban environments. Targeting the operational needs of municipal authorities and marketing service agencies, SCOUT integrates rule-based heuristics, large language models (LLMs), and machine learning (ML) techniques, including a custom-trained YOLOv11 object detector. The system processes dashcam video streams to identify advertisement banners and automatically logs their locations in a spatial database using GPS metadata. Comparative experiments evaluate the performance of traditional computer vision methods, LLM-based analysis, and deep learning models, highlighting the limitations of classical and LLM-based approaches in dynamic urban settings. Results demonstrate that the hybrid system - anchored by a lightweight object detection model - offers significant improvements in detection accuracy, scalability, and real-time applicability. SCOUT provides a novel, practical solution for urban advertisement monitoring and policy enforcement, with potential for integration into larger smart city infrastructure through third-party APIs.

1. Introduction

Manual inspection of outdoor advertisement placements is a time-consuming and labor-intensive process that often involves field personnel physically surveying city streets to identify banner locations, assess visibility, and detect any damage or regulatory violations. This approach not only incurs high operational costs but also lacks the agility required for timely response, particularly when banners are damaged, missing, or incorrectly placed. For advertising agencies operating in competitive and fast-paced markets, delays in detecting such issues can result in lost impressions, reputational damage, and unmet client expectations. There is a critical need for an automated, real-time system that enables rapid identification and verification of banner placements, allowing agencies to swiftly respond to defects or compliance issues and maintain service quality (Azimi et al., 2018; Zhou et al., 2022).

The introduced Street-level Capture & Overlay for Urban Track-inspection (SCOUT) is an automation that allows the user to track outdoor advertising banners using high-resolution video captured during street-level driving. The project fuses object detection and geospatial mapping technology, thus adopting a more systematic approach to monitor and control the distribution of outdoor advertising across the city (Zhou et al.,

Received 17 March 2025, Received in revised form 19 May 2025, Accepted 2 June 2025
http://doi.org/10.25045/jpit.v16.i2.01

2022). Every detected banner is indicated on a digital map, the location of which is extracted from the GPS metadata of the video frames. Together, these components form a complete pipeline for the capture, geolocation, and visualization of banner data—presenting it to both the public and municipal authorities. This system contributes not only to improving regulatory enforcement but also to enhancing urban aesthetics while fostering licensed commerce within the locality (Gonzalez et al., 2019).

The development of the SCOUT system progressed through multiple stages, each involving distinct strategies aimed at improving the accuracy of outdoor banner detection. In the initial phase, we employed traditional algorithmic approaches from computer vision, including Hough transformations and edge detection (Canny, 1986), to identify banners modeled as large rectangular structures. While these methods yielded promising results under controlled conditions—particularly when parameters were finely tuned—they proved highly sensitive to environmental variability such as lighting and occlusion. The lack of robustness led to a high incidence of false positives and suboptimal detection accuracy, necessitating the transition to more advanced learning-based techniques (Dalal & Triggs, 2005).

Given the emergence of Large Language Models (LLMs), we explored their potential applicability to vision-language tasks. Integration trials were conducted using ChatGPT and LLaMA2 via API-level prompting to analyze images and identify advertisement banners. Although LLMs demonstrated some capacity for visual reasoning (Gadre et al., 2023; Huang et al., 2023), their deployment in this context was hindered by practical limitations, including high computational overhead, limited visual specialization, and challenges in pipeline integration (Zhao et al., 2023).

The final implementation of the SCOUT system centered around the YOLOv11 architecture, which builds upon earlier work in real-time object detection. The model was trained on a custom dataset of 341 manually annotated images containing advertisement banners. After a comprehensive tuning process, the YOLOv11 model achieved consistent detection results across diverse video streams. Each recognized banner instance was mapped to video metadata, and the corresponding GPS coordinates were extracted and stored in a spatial database for automated geolocation and mapping.

By integrating artificial intelligence with geospatial technologies, SCOUT facilitates the scalable and automated monitoring of outdoor advertisements in urban settings. The system enables efficient oversight of banner placements and supports regulatory compliance with municipal advertising codes (Zhou et al., 2022). Despite the modest size of the training dataset, the model demonstrated generalization capabilities under varying lighting and occlusion conditions—suggesting that SCOUT can be effectively adapted to other urban regions with minimal retraining effort.

In sum, SCOUT exemplifies the practical application of AI-powered object detection integrated with geospatial mapping for urban governance. The system provides an operational viable solution for balancing commercial advertising activities with the public interest in city design and regulation. The following sections detail the technical architecture of the system and report on experimental evaluations under real-world conditions.

2. Overview of the system

This section describes the components of the system, their interaction and effects of using alternatives in the building blocks. The general idea of the SCOUT system, which will be referenced throughout this section is shown in Fig. 1. This section also evaluates and discusses the effect of different solutions.

2.1. Rule-based strategy with classical algorithms.

A range of rule-based methods for detecting advertisement banners was explored through classical computer vision techniques. Among these, Hough Transform was initially employed in conjunction with line detection and rectangular shape recognition to identify potential advertisement panels within video frames. The rationale behind this approach assumed that advertisement banners typically exhibit strong linear features and well-defined rectangular boundaries, making them suitable candidates for detection through geometric primitives.

The first step in this method involved applying the Hough Transform to locate prominent edges within each frame.

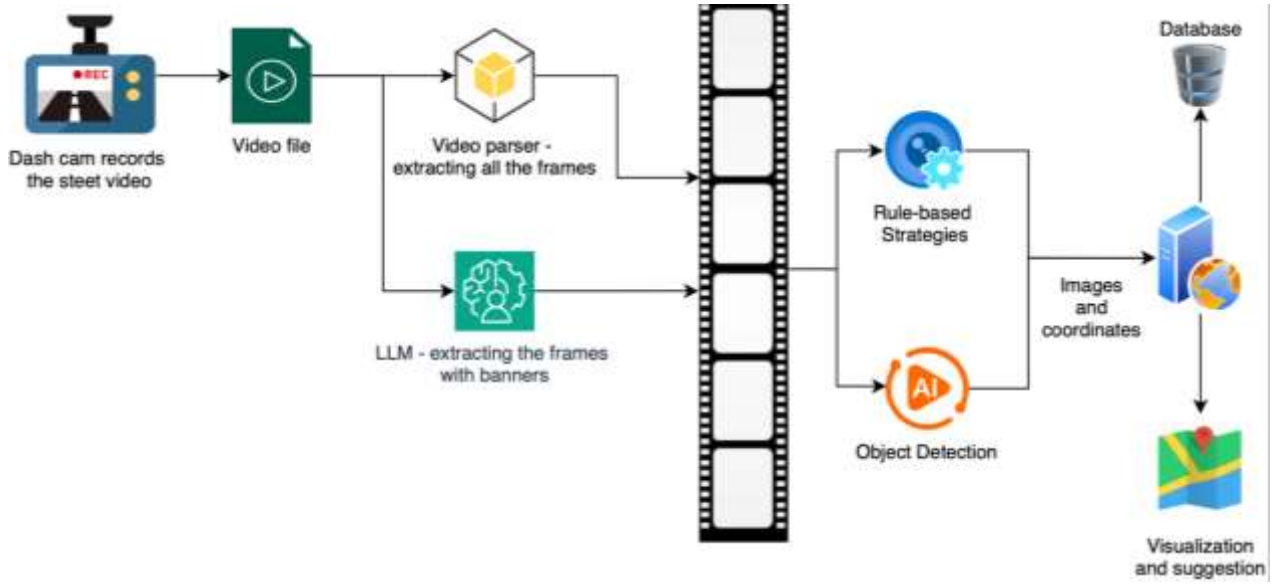

Fig 1. The general overview of the system

Focus was given to the upper-right quadrant of the frame, based on domain knowledge that suggested advertisements were frequently positioned in this region of the captured scenes. The objective was to detect linear segments that could indicate the presence of banner edges, especially those corresponding to billboard or panel boundaries.

However, in practice, this technique revealed significant limitations. The edge-detection process using the Hough Transform frequently identified spurious or irrelevant lines—many of which stemmed from background elements, environmental noise, or architectural features unrelated to advertisements. These false positives introduced a high level of ambiguity into the system, making it difficult to reliably isolate the actual banners. The variability in lighting conditions, viewing angles, and occlusions further degraded the robustness of this method.

Given these challenges, the effectiveness of using Hough Transform in isolation was deemed insufficient for accurate banner detection. The high rate of false detections and the method's sensitivity to visual variance indicated that a more robust, data-driven approach would be necessary. As a result, the decision was made to discontinue further refinement of rule-based geometric detection and shift towards more adaptive and learning-based techniques that could better accommodate the complexities of real-world imagery. Fig. 2 shows an example of Hough transform-based detection.

To improve the reliability of line detection in the context of advertisement banner identification, specific parameter constraints were introduced into the Hough Line Transform algorithm. One of the primary strategies to eliminate irrelevant or spurious lines involved filtering based on the length of the detected lines. In particular, only lines whose lengths fell within 20–25% of the image height were considered candidates for banner boundaries. This thresholding approach was grounded in the observation that advertisement banners often span a significant portion of the image width or height and exhibit long, well-defined edges. By discarding shorter lines, which are more likely to originate from background clutter, environmental structures, or texture noise, the detection algorithm aimed to reduce false positives and improve focus on structurally meaningful elements.

Fig. 2. An example of results of Hough Transform

In addition to length-based filtering, the search was geographically constrained to the top-right quadrant of each image frame, as prior empirical

observations revealed that this region frequently contained outdoor banners in the dataset under consideration. This spatial restriction served as a heuristic to reduce the computational load and narrow down the detection scope. While this method contributed to a notable improvement in detection precision, it came at the expense of processing efficiency, causing a fourfold increase in computational time due to the intensive filtering and validation steps introduced.

To further refine edge detection prior to applying the Hough Transform, a Canny Edge Detection filter was integrated into the preprocessing pipeline. The Canny filter effectively enhances edge localization by suppressing noise and isolating regions of strong intensity gradients—features that are essential for delineating banner boundaries with higher structural clarity. This step is particularly valuable in urban imagery, where shadows, reflections, and textured surfaces can confound simpler edge detection methods.

However, despite these refinements, tuning the parameters of both the Canny and Hough algorithms remained a nontrivial task. The Canny filter relies on two threshold parameters (low and high), which must be carefully balanced to avoid missing weak edges or introducing excessive noise. Similarly, Hough Transform's performance is sensitive to its accumulator threshold, minimum line length, and maximum line gap settings. In our case, the minimum line length was explicitly calibrated to fall within 20–25% of the image height, and the maximum gap between line segments was kept small to ensure line continuity. These parameters must be adjusted in tandem with the resolution and scene complexity of the input video frames, and slight deviations often led to a dramatic increase in false detections or missed banners.

Ultimately, while these enhancements improved the specificity of rule-based line detection, they still fell short of the robustness required for consistent banner identification across varied urban scenes. This highlighted the inherent limitations of purely algorithmic approaches and motivated a transition toward more adaptive, learning-based models.

2.2. LLM-based detection

Recent developments in artificial intelligence have made it feasible to integrate Large Language Models (LLMs) with traditional object detection frameworks, enabling systems to go beyond visual recognition and provide interpretive or contextualized outputs. This integration has been explored in recent research where object detectors, such as YOLOv8, are combined with LLMs like LLaMA2 to enhance performance in tasks requiring both visual and linguistic reasoning.

In one notable implementation, YOLOv8 was used for detecting objects—in this case, advertising banners—while LLaMA2 was tasked with interpreting or responding to the detected outputs. The integration was accomplished using a model fusion approach, where the output of the vision model was translated into descriptive sentences and subsequently processed by the LLM to generate appropriate responses (Wase et al., 2023). This architecture avoids the complexity of traditional vision transformers by adopting a lightweight fusion layer, while still achieving strong performance—reportedly up to 88.16% accuracy in test scenarios relevant to autonomous driving and real-time object detection.

Beyond this example, broader research suggests that such vision-language fusion frameworks are well-suited for use in robotics, surveillance, and human-computer interaction scenarios. For instance, systems like VisionLLM treat images as sequences of language tokens, enabling LLMs to function as flexible decoders for vision-centric tasks across varied domains (Wang et al., 2023). Similarly, survey work on LLMs in robotics outlines the potential and challenges of using such models to improve situational awareness and autonomous decision-making (Wang et al., 2024).

In the search of higher accuracy, better response time and candidacy for the possible ensemble design, alternative approaches were explored to improve object detection performance and reduce computational demands. Among these, the **Segment Anything Model (SAM)** and **Small Language Models (SLMs)** were considered. SLMs require less pre-trained data and offer lightweight inference capabilities; however, they generally deliver lower accuracy compared to large-scale models such as YOLO and Large Language Models (LLMs). Consequently, these alternatives were deemed less effective for the precise and high-frequency detection tasks required by the SCOUT system.

The **SAM** was also integrated as part of a preliminary testing phase.

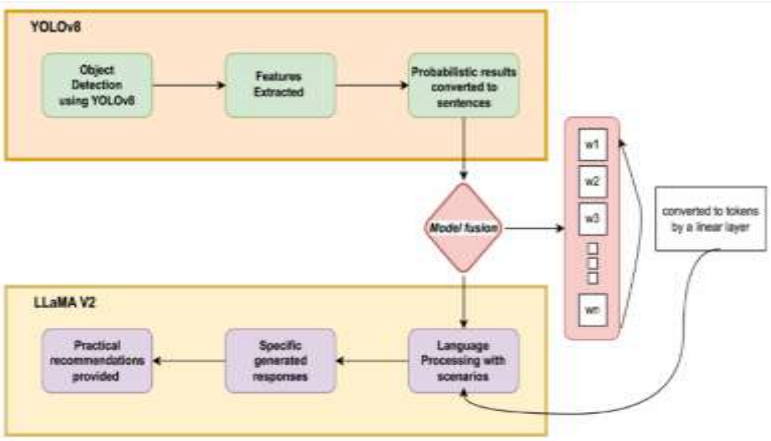

Fig. 3. A combination of YOLO and LLM model.

Although SAM demonstrated the ability to track objects persistently across video sequences - successfully monitoring banner transitions from one segment to another - it was significantly limited in scope. Specifically, the demo version of the model could only track a single object per session. When multiple banners were present in a frame, the model prioritized one and ignored the others, which made it unsuitable for comprehensive detection scenarios (Jiaowoguanren, 2022).

In addition to functional limitations, SAM posed technical integration challenges. Implementing **FFMPEG** in conjunction with the SAM pipeline led to operational incompatibilities, particularly with **CUDA-based operations**. This incompatibility affected the stability and efficiency of the video processing pipeline, further disqualifying SAM as a viable option for real-time detection within the SCOUT system. Due to the mentioned limitations - restricted multi-object detection, inefficient handling of GPU resources, and integration difficulties - SAM and related lightweight alternatives were ultimately excluded in favor of the more performant and scalable YOLOv11-based detection approach.

In addition, the usability of the SAM2 model was evaluated. This model leverages several advanced techniques, including zero-shot segmentation based on the SAM2 framework, autoregressive mask propagation for temporal consistency, and mixed precision inference to optimize GPU resource utilization. Furthermore, it employs binary mask processing via OpenCV for efficient storage and management of segmented outputs. The details of the implementation are reflected in Algorithm 1.

Despite its latency, which due to the specifics of the business was not the primary task of the SCOUT, it was decided to embed the LLM model to detect the presence of the banner in the given frame or detect the frames in the video that contains banners. This step was assumed to increase the inference time but narrow down the target to improve the accuracy.

Algorithm 1 SAM2 Inference for Banner Segmentation

Init: Enable AMP: `torch.autocast("cuda", bfloat16)`
Initialize predictor: $P \leftarrow \texttt{init}(V_{path})$
Define bbox: $B \leftarrow [x_1, y_1, x_2, y_2]$
Prompts: $p \leftarrow \texttt{genPrompts}(B)$
$[ID_0, M_0] \leftarrow P.\texttt{add}(p)$
for $f \in V$ **do**
  $[ID_f, M_f] \leftarrow P.\texttt{propagate}(f)$
  $\texttt{print}(f.\texttt{idx}, ID_f)$
  $F \leftarrow \texttt{format}(\text{"}mask\_\%d.png\text{"}, f.\texttt{idx})$
  $M'_f \leftarrow \texttt{toBinary}(M_f[0])$
  $\texttt{save}(\text{F}, M'_f)$
end for

In the experiment, both the free version of ChatGPT (featuring GPT-4o trials) and the paid version of ChatGPT-4 were used to evaluate their effectiveness in detecting advertisement banners within images. For this evaluation, a standardized test image (as shown in Fig. 4) was provided, along with the prompt: *"Is there any advertisement banner in this image?"*

Fig. 4. An example of image input to the LLM.

Despite variations in the placement of banners within the image, both GPT-based models demonstrated comparable success in accurately identifying the presence of advertisement content. Notably, even when part of the banner was visually occluded - such as by a person standing in front of it - the models were still capable of correctly detecting the banner and its promotional content.

A representative response from the GPT model was as follows: *"Yes, there is an advertisement banner in the image. It is located on the right side, attached to a pole, promoting a place called 'Sirr Cake House.' The banner features an image of cakes or desserts, along with contact information."*

To facilitate the input image processing, a Python-based application was developed. This application encoded the input image in Base64 format and submitted it, along with a textual prompt, via an API call to OpenAI's platform. This method offered a lightweight and practical alternative to building a dedicated banner detection model. The API key required for access was generated through the OpenAI API key management.

Furthermore, API usage was governed by rate limits based on the user's subscription level, as outlined on the OpenAI usage settings page. A list of the available GPT models and their specifications is provided in Table 1.

To evaluate the feasibility of GPT-based models in advertisement banner detection, four different models were tested using a Python-based application. The experiment utilized three-second frame captures extracted from a one-minute Baku dashcam video. These frames were processed via the OpenAI API, using both direct image links and base64-encoded formats as input.

Initial trials revealed significant limitations. While access to GPT-4 was granted through a paid API token, some models such as GPT-4 Turbo remained inaccessible, despite the subscription. During testing, attempts to process images using both URL references (e.g., via Imgur) and base64 encoding resulted in repeated failures. The GPT-4 model returned general responses indicating an inability to analyze image data through URL or base64 methods. In some cases, input exceeded token-per-minute (TPM) limits, resulting in processing errors. For example, attempts to process base64 images frequently surpassed the 10,000 TPM limit imposed by OpenAI's API, halting further computation.

Efforts to circumvent these restrictions by switching to the GPT-4o mode, which offers extended token capacity also proved ineffective. Although system throughput improved marginally, image-based prompts were still not processed successfully. This suggests that despite theoretical support for multimodal capabilities, practical access to vision-enabled models is not guaranteed across all API tiers or account configurations.

Table 1. List of available models.

Model	Token Limits (TPM)	Request Limits (RPM / RPD)	Batch Queue Limits (TPD)
gpt-3.5-turbo	200,000	500 / 10,000	2,000,000
gpt-3.5-turbo-0125	200,000	500 / 10,000	2,000,000
gpt-3.5-turbo-1106	200,000	500 / 10,000	2,000,000
gpt-3.5-turbo-16k	200,000	500 / 10,000	2,000,000
gpt-3.5-turbo-instruct	90,000	3,500 / –	200,000
gpt-3.5-turbo-instruct-0914	90,000	3,500 / –	200,000
gpt-4	10,000	500 / 10,000	100,000
gpt-4-0613	10,000	500 / 10,000	100,000

Further investigation confirmed that access to GPT-4 Vision was not available under the tested OpenAI account, and that image transmission must occur through direct upload rather than URL or base64 input. Given that OpenAI's API currently restricts image upload capabilities, the method proved unsuitable for real-time or cost-efficient integration.

In summary, integrating GPT models into the advertisement banner detection pipeline introduced multiple technical and operational challenges. Token limitations, model accessibility constraints, and inefficient processing of non-informative frames (i.e., those without banners) significantly hindered system performance. Since a large proportion of dashcam footage lacks banner content, continuous frame processing resulted in excessive computational cost and token waste. These factors ultimately led to the abandonment of

GPT models in favor of more tailored and efficient detection solutions.

3. Data collection

For machine learning (ML)-based strategies, the availability of a suitable training dataset is a critical requirement. Fortunately, the application of transfer learning on large pre-trained models significantly reduces the need for extensive data, allowing effective training with relatively smaller sample sizes. In this study, a dataset comprising approximately 400 images of advertising banners was collected from various locations throughout Baku city. These samples were extracted from both still images and video footage.

We employed the YOLOv11 model within a transfer learning framework, which demonstrated stable and consistent performance during testing. Given the modest size of the dataset, the model was trained for 50 epochs to mitigate the risk of overfitting. This training regime yielded favorable results, with the system successfully detecting between 70% and 90% of the banners in test videos.

Beyond the core AI model, the system also incorporates a geospatial mapping component. Leveraging metadata extracted from the collected media, the system maps detected banners to specific locations using embedded geolocation information. This enables a visual representation of banner distribution across the city, enhancing the system's practical utility for urban monitoring and advertisement analysis.

4. Experimental results

To begin the experimentation phase, the Ultralytics repository was configured to run the latest available version of YOLO - YOLOv11. Various task variants offered by the model, including object detection, instance segmentation, keypoint detection (pose estimation), oriented object detection, and image classification, were systematically evaluated using both static images and video input.

During initial tests, YOLOv11 successfully identified common traffic elements such as traffic lights and stop signs. However, it consistently failed to detect advertisement banners. This limitation was attributed to the fact that YOLOv11, in its default configuration, is pre-trained on the COCO dataset, which does not include advertisement banners as labeled classes. This outcome led to the conclusion that fine-tuning YOLOv11 on a custom dataset tailored for advertisement banner detection would be necessary.

To broaden the evaluation, additional Single Shot Detector (SSD) models were implemented, including ResNet50, ResNet152, and VGG16 (sourced from the PyTorch Torchvision library), as well as MobileNetSSD (from OpenCV). Among these, the PyTorch-based models produced limited results, occasionally mislabeling electronic banners as 'TVs.' MobileNetSSD failed to detect any banners. These results further reinforced the necessity of training a specialized model for advertisement banner recognition.

A custom dataset was subsequently created using 16 GB of dashcam footage. Still frames containing banners were extracted and uploaded to Roboflow, where a new project was established under the “Object Detection” task type, with a single annotation group labeled “banner.” Manual annotation was performed on all banner-containing frames. The dataset was then exported, and training was executed in a Google Colab environment. Using the Ultralytics framework, the dataset was configured in a data.yaml file, and the runtime was switched to a T4 GPU to expedite training. Following completion, the trained model weights (best.pt) were downloaded for further testing.

Model validation was carried out using a custom Python script, which analyzed video inputs frame by frame and applied bounding boxes to detected banners. The initial evaluation on 30 labeled frames yielded satisfactory results, prompting an expansion of the dataset to 100, 160, and finally 341 annotated images.

Further refinements were implemented to improve detection utility. For each detected banner, the bounding box with the largest area was selected to ensure higher visual resolution. Metadata including video ID, image path, and estimated GPS coordinates (where available) were stored in a structured database.

To accurately determine the geolocation of detected banners, an investigation into GPS-enabled video metadata was conducted. Analysis of the existing footage using ExifTool revealed that no embedded GPS information was present. Attempts to use mobile phones for recording proved insufficient, as most recorded GPS only at the video start. Although action cameras such as the GoPro HERO5 Black offer continuous GPS

metadata logging, their cost (~$450) necessitated alternative solutions.

Two mobile applications from the Android Play Store were evaluated for suitability. The first, *GPS Logger*, allowed second-by-second location recording and exported the data in KML or GPX format. The second, *Timestamp Camera Free*, provided a more integrated approach by overlaying real-time GPS coordinates and timestamp data directly onto the video. This application was chosen due to its superior usability and seamless recording functionality.

Corresponding modifications were made to the Python-based detection pipeline. Using the Pytesseract library, optical character recognition (OCR) was applied to the region of each frame containing the detected banner. High-resolution frames were stored in a best_frames array, and a second processing loop was employed to extract text data from these frames. A regular expression pattern was designed to identify and extract GPS coordinates from the detected text.

Videos recorded using *Timestamp Camera Free* were subsequently processed. In approximately 10–15% of cases, GPS text was not correctly detected, due to the overlay's positioning. To resolve this, a targeted cropping operation was applied to isolate the bottom-right corner of each frame—where the GPS text was rendered. This modification resulted in successful GPS text extraction in all remaining cases.

Although it is hard to compare the performance of the mentioned methods, which is reasoned with the lack of baseline data and absence of a banner as a class in the object detection datasets, the authors used their own labeled dataset for the evaluation. Comparison of the performances of different approaches is made based on the Accuracy and Precision metrics:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

where TP (True Positive) refers to correctly identified positive instances, such as correctly detected objects; TN (True Negative) denotes correctly identified negative instances, meaning non-objects that were correctly ignored; FP (False Positive) represents instances incorrectly identified as positive, such as detecting an object where there is none; and FN (False Negative) refers to missed positive instances, where an actual object was present but not detected.

Table 2 provides the performance metrics of different approaches.

Table 2. Comparison of the functional performance of different approaches

Approach	Accuracy	Precision
Classic Computer Vision Algorithms	50%	73%
LLM (GPT-4)	64%	68%
LLM (Llama)	61%	66%
YOLOv11	92%	89%

Traditional computer vision algorithms struggled with rectangular shapes, especially on buildings and traffic signs, leading to low precision due to false detections. Large Language Models like GPT-4 and LLaMA, although promising in multimodal tasks, also misclassified rectangular regions such as signs and banners, highlighting their limited capability in pure vision tasks through current APIs. Among the tested methods, YOLOv11 delivered the most promising results due to its robust architecture and ease of training and testing. It achieved up to 90% accuracy in ideal conditions and maintained around 85% in real-world video tests, although it still faced challenges with rectangular structures. Overall, while YOLOv11 stands out as the best-performing model, all methods revealed the inherent difficulty of distinguishing semantically meaningful rectangles in urban imagery.

5. Web Application Implementation

The SCOUT system has been developed as a web application that enables video file upload and provides interactive visualization of detected banners on a geographic map (see Fig. 5). The SCOUT web application offers a seamless and interactive user experience, combining video processing capabilities with geospatial visualization to support banner detection and exploration.

The web interface serves as the second major component of the SCOUT application. It enables users to upload videos and observe real-time detection results as the processing progresses (see Fig. 5). Once the processing is complete, users can navigate to a map-based visualization module, where banner locations are displayed dynamically using the Leaflet.js library. This interactive map is generated based on the GPS coordinates stored in the database, and users can click on location

markers to view the corresponding cropped banner images (see Fig. 5).

Fig. 5. Banner detection preview.

The web application components are tightly integrated via the PostgreSQL database. Part 1 of the system handles the video analysis and data storage, while Part 2 retrieves the stored data using video-specific identifiers to present it in the web interface. The Flask framework acts as the middleware, enabling communication between the front-end interface, video processing backend, and database layer. RESTful APIs are used to fetch detection results and serve them to the client-side application, ensuring efficient and dynamic rendering of the map and associated banner information.

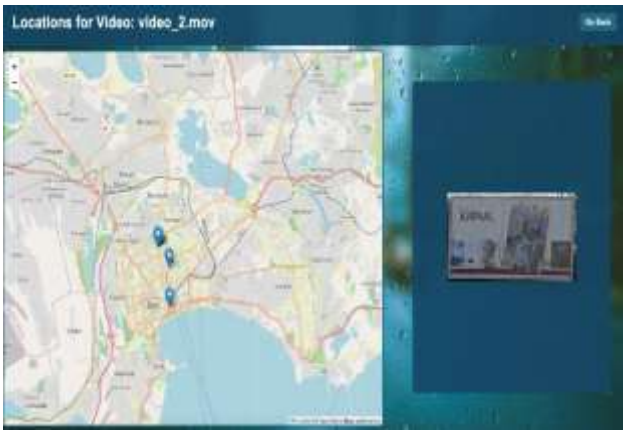

Fig. 6. Example of locations being displayed.

5.1. Integration challenges and solutions

One of the primary technical challenges during the development of the SCOUT system was the real-time integration of banner detection with video display functionality. To maintain the objective of real-time interactivity, the application needed to simultaneously process video frames using the YOLOv11 model and render the detection results—bounding boxes and associated labels—on the video stream without any perceivable performance degradation.

The computational demands of the YOLOv11 model posed a major obstacle, as it significantly delayed video stream operations when processing every frame. To achieve real-time performance, it was necessary to ensure that the frame-by-frame inference and video playback operations were precisely coordinated within a single application context. Without proper synchronization and optimization, full-frame analysis risked compromising the fluidity of the video stream.

To address this issue, we embedded the frame processing logic into the generate_stream function within the Flask application. This function utilizes OpenCV's cv2.VideoCapture to read frames from the uploaded video. Each frame is then processed by the YOLOv11 model to detect and label banners. The processed frames are modified by drawing bounding boxes and labels over the detected banners and are subsequently JPEG-encoded.

Using Flask's streaming capabilities with Python generators (via the yield function), the application sends these processed frames as a continuous stream to the client's web browser. This design allows for a steady and responsive video stream while preserving real-time detection capabilities. Further performance improvements were achieved by resizing each frame to 1600×896 pixels before processing. This resolution provided a balance between detection accuracy and computational efficiency. The use of lightweight frame encoding and streaming with Flask and OpenCV ensured that the user experience remained smooth even under real-time conditions.

These architectural refinements, selective frame scaling, and careful coordination of the detection and streaming tasks within the Flask application framework made the promised integration of real-time detection and visualization possible.

6. Conclusion and future work

This research demonstrated the feasibility and utility of an AI-powered banner detection and geospatial mapping system tailored specifically for the urban environment of Baku. The developed system integrates object detection and geolocation technologies to identify outdoor advertisement banners in dashcam footage and map them in real time. Central to the system is a custom-trained YOLOv11 model, fine-tuned on a

dataset of over 400 manually labeled images. The training process, configured to run for 50 epochs, was optimized to maintain generalizability while mitigating the risk of overfitting.

Experimental results confirmed that the system performs robustly under varied environmental conditions, including differing lighting and partial occlusions, with detection accuracy ranging from 70% to 90%. Crucially, each detection instance is automatically logged in a spatial database, enabling a real-time, map-based user interface that reflects the current distribution of urban advertisements. This spatial integration not only supports data visualization but also facilitates policy enforcement and urban planning workflows.

The study's transition from classical image processing techniques—such as Hough transform and Canny edge detection—and initial explorations using large language models (LLMs) like ChatGPT and LLaMa2, to a purpose-trained object detection model, proved decisive. Classical approaches were highly susceptible to false positives and sensitive to environmental noise, while LLM-based vision processing was prohibitively costly and inefficient for real-time, city-scale deployment. In contrast, the YOLOv11 architecture offered a lightweight, efficient, and scalable solution for continuous video stream analysis either at the edge or in the cloud.

What distinguishes the proposed system is its seamless integration of visual AI with geographic information systems (GIS). From frame extraction and object detection to geocoding and database population, the system ensures a fully automated workflow. This makes it a powerful tool for city administrators and regulatory bodies, offering transparent, up-to-date oversight of outdoor advertising infrastructure. The system supports not only regulatory enforcement and commercial inventory management but also contributes to more human-centered urban design by identifying and curating visual clutter in public spaces.

Future enhancements could include augmenting the training dataset with images representing various seasonal and weather conditions to further improve detection robustness. The incorporation of semantic segmentation methods may allow for more granular attribute extraction, such as banner dimensions, orientation, and textual content. Additionally, real-time alert modules, integrated with web or mobile dashboards, could enable timely responses to unauthorized or non-compliant advertising installations.

Finally, the system architecture supports potential integration with third-party APIs, enabling regulatory organizations to track advertising banners more effectively over time. While the current solution is not yet scalable to a global level, it presents a working prototype of a smart urban monitoring system that could serve as a model for other cities aiming to enhance control, compliance, and quality of their outdoor advertising landscape.

Acknowledgement

This work was supported by the Science Foundation of the State Oil Company of Azerbaijan Republic (SOCAR) (Contract No. 01LR-EF/2024). The Proof-of-Concept model designed on SCOUT has been used by *Adviad* and *NewMedia* advertisement agencies.

Funding Support

No funding information is available.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

The data that support the findings of this study are openly available at https://github.com/eljanabdullazada/sdp_cv

References

Azimi, S. M., Vig, E., Bahmanyar, R., Körner, M., & Reinartz, P. (2018). Towards multi-class object detection in unconstrained remote sensing imagery. Proceedings of the European Conference on Computer Vision (ECCV).

Zhou, P., Lin, Z., & Liu, Y. (2022). GeoAI: Integrating AI and geospatial analytics for smart cities. ISPRS International Journal of Geo-Information, 11(2), 107.

Gonzalez, R. C., Woods, R. E., & Eddins, S. L. (2019). Digital Image Processing Using MATLAB. Pearson Education.

Canny, J. (1986). A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, (6), 679–698.

Dalal, N., & Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR).

Gadre, S., Misra, I., Rohrbach, A., Zitnick, C. L., & Achlioptas, P. (2023). CLIP the Gap: A Single Modality Representation Learning for Vision-Language Tasks. CVPR 2023.

Duda, R. O., & Hart, P. E. (1972). Use of the Hough transformation to detect lines and curves in pictures. Communications of the ACM, 15(1), 11–15. https://doi.org/10.1145/361237.361242

Jiaowoguanren. (2022, August 15). Billboards-signs-and-branding TF ResMLP. Kaggle. https://www.kaggle.com/code/jiaowoguanren/billboards-signs-and-branding-tf-resmlp

Naeem, M., Asim, M., & Ali, M. (2023). I2MVFormer: Large language model generated multi-view document supervision for zero-shot image classification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 1–9). IEEE CVPR 2023 Open Access Repository.

OpenAI. (n.d.). Rate limits. OpenAI. Retrieved April 28, 2025, from https://platform.openai.com/settings/organization/limits

Wase, Z. M., Madisetti, V. K., & Bahga, A. (2023). Object detection meets LLMS: model fusion for safety and security. Journal of Software Engineering and Applications, 16(12), 672–684. https://doi.org/10.4236/jsea.2023.1612034

Wang, W., Chen, Z., Chen, X., Wu, J., Zhu, X., Zeng, G., & Dai, J. (2023). VisionLLM: Large Language Model is also an Open-Ended Decoder for Vision-Centric Tasks. arXiv preprint arXiv:2305.11175. https://arxiv.org/abs/2305.11175

Wang, J., Wu, Z., Li, Y., Jiang, H., Shu, P., Shi, E., & Zhang, S. (2024). Large Language Models for Robotics: Opportunities, Challenges, and Perspectives. Journal of Automation and Intelligence. https://doi.org/10.1016/j.jai.2024.12.003