AI-Assisted Object Detection System for Security Purposes in Educational Facilities using Python and Computer Vision (CV) Libraries

Tshikala[1], N'Zi Ngenda[2], Eusaoad[3], Bhattarai[4] and Habib[5*]

[1,2,3] *Department of Computer Engineering, Assumption University, Bangkok, THAILAND*

[4,5] *Department of Electrical and Computer Engineering, Assumption University, Bangkok, THAILAND*

**Corresponding author; E-mail address: shabib@au.edu*

Abstract

As artificial intelligence (AI) continues to reshape engineering education, integrating real-world AI applications into the undergraduate curriculum has become essential. This paper presents a student-led implementation of an AI-assisted object detection system designed for enhancing safety in educational facilities. Developed as a capstone project, the system demonstrates the practical application of Python-based computer vision libraries—YOLO, OpenCV, TensorFlow—alongside database tools and a custom graphical user interface (GUI). It identifies and classifies objects such as bags, bottles, electronic devices, and potential hazards (e.g., knives, scissors) in real-time, linking detections to student profiles through a Django-MySQL/SQLite backend. The GUI, developed using HTML, CSS, and JavaScript, enables administrative monitoring and data analysis. Beyond its security function, the project exemplifies interdisciplinary skill development in AI, software engineering, and data systems. This work highlights the pedagogical value of AI-driven project-based learning in engineering programs, promoting hands-on experience, ethical considerations, and systems-level thinking among students preparing for the future of intelligent infrastructure.

Keywords: AI in education, object detection, computer vision, engineering education, project-based learning

1. Introduction

Educational institutions have a mandate to provide a learning environment which ensures the well-being of students, staff members, and facilities in an atmosphere that is safe and secure for all. Ensuring the security of educational facilities can be challenging due to growing student populations, large campuses, and significant security threats in the contemporary society [1]. Traditional methods involving surveillance cameras, manual monitoring, and access control systems are susceptible to inefficiencies in the complete detection and response to potential dangers. Recent breakthroughs in machine learning and computer vision have transformed the realm of automated surveillance, where systems can analyze visual data in real-time and detect specific objects or threats with high accuracy [2]. However, a great number of educational facilities still depend on either antiquated or manual systems, which introduces inefficiencies and security vulnerabilities in the operation. Consequently, there is a need for better security protocols in educational institutions through the installation of modern, intelligent systems that can help overcome various prevalent challenges, such as real-time [3] threat identification. Such systems could deploy intelligent object detection and classification, and automatically reduce the time spent detecting and responding to perceived threats. Effective automation also minimizes reliance on human operators, reducing fatigue and other errors associated with manual monitoring. Data-driven insights from detection logs could assist in better decision-making and future security planning. Such intelligent systems could also be made scalable to different campus sizes and designed for specific security requirements, thus making them fit for different educational settings.

With the above concerns in mind, our project aims to create a smart security system that improves safety in educational institutions using object detection and classification. By leveraging machine learning models like YOLO (You Only Look Once) or Convolutional Neural Network (CNN), the system can quickly and accurately identify objects such as weapons, unauthorized items, or suspicious activities in real-time. The proposed work focuses on creating an AI-assisted security system to improve safety in educational institutions. It can work with existing CCTV cameras, process live video feeds, and alert security personnel instantly when a risk is detected. Designed to be scalable and adaptable, the system can be enhanced in future with advanced features like facial recognition and behavior analysis for even stronger security.

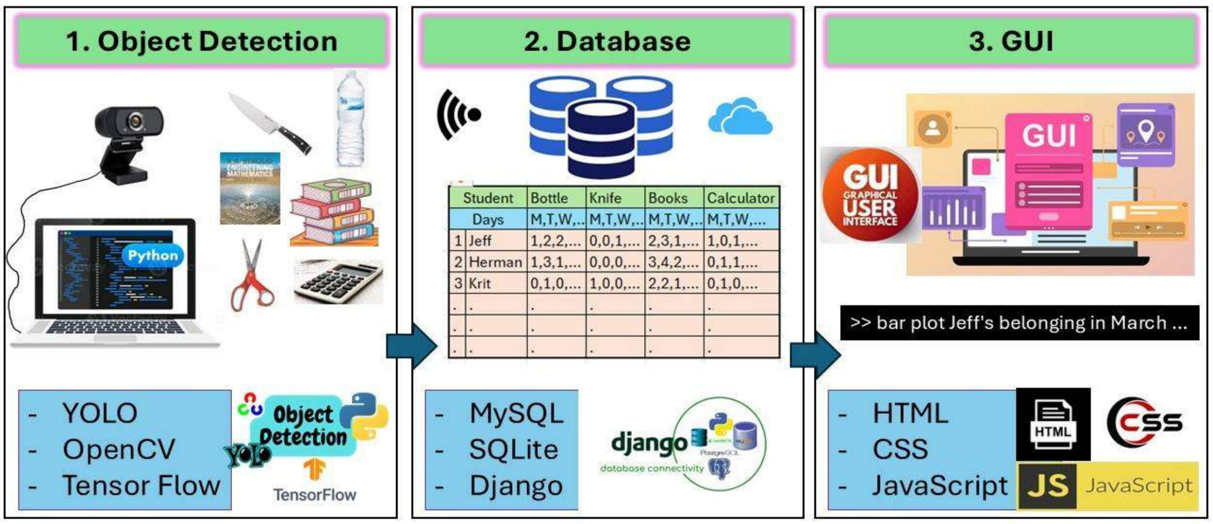
2. Overview of Existing Systems

Object detection and classifications have evolved from traditional function-based techniques to deep learning-based models, which improve the accuracy and efficiency of applications in the real world. Previous methods depended on Support Vector Machines (SVM) [4] as functional extraction algorithms such as Histograms of Oriented Gradients (HOG) [5], which were usually used to detect pedestrians. Other approaches, such as the Scale Invariant Feature Transformation (SIFT) and Speeded-Up Robust Features (SURF) [6], were effective for the recognition of objects but struggled with effectively identifying animals. The Viola-Jones algorithm was the first successful facial detection method, which utilizes Haar-like features to identify objects in images. With progress in deep learning, the accuracy and speed of object detection have improved significantly. Fast R-CNN, Faster R-CNN, and Mask R-CNN – all of which belong to the Region-based Convolutional Neural Network (R-CNN) family, introduced regional proposal techniques for accurate object location [7, 8]. Meanwhile, YOLO and Single Shot Multibox Detector (SSD) provide real-time object detection, making them widely used in applications that require speed such as autonomous vehicles and safety systems [9]. Object detection using transformer-based architectures leverages a self-attention mechanism, as seen in recent developments like Detection Transformers, to improve the accuracy of object recognition [10]. In addition, light models such as Mobilate and Edge AI solutions enable the object detection of mobile and built-in devices and improve access and distribution in low-power environments. As the research continues, the object classification technique is expected to further enhance object classification technology through self-supervised learning and integration of Vision Transformer (VIT).

3. Proposed System Model

System architecture plays a fundamental role in the design of efficient and scalable systems by defining their structure, components, and interactions. Traditional monolithic architecture, where all system components are closely integrated, was widely used in early software development, but often faced challenges in scalability and maintenance. To address these boundaries, client-server architecture appeared and shared the responsibility between a central server handling data and several clients managing business logic and user interactions [11].

Our proposed system consists of three main parts and associated processes as illustrated in Fig. 1 below, which outlines the major steps involved from data acquisition to result storage.



	Student	Bottle	Knife	Books	Calculator
	Days	M,T,W,...	M,T,W,...	M,T,W,...	M,T,W,...
1	Jeff	1,2,2,...	0,0,1,...	2,3,1,...	1,0,1,...
2	Herman	1,3,1,...	0,0,0,...	3,4,2,...	0,1,1,...
3	Krit	0,1,0,...	1,0,0,...	2,2,1,...	0,1,0,...
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

Fig. 1 Overview of the Proposed System

The aforementioned structured process flow ensures accurate, efficient, and scalable object detection, making it suitable for real-time applications. The workflow of the system is described below:

3.1 Object Detection

A USB camera connected to a laptop computer acquires images of objects inside a student's bag. These images are fed into the object detection Python libraries using YOLO, OpenCV, and TensorFlow.

3.2 Database

Once the object(s) are identified, the information is stored in a database using SQL and Django. As shown in Fig. 1, name/id of the person carrying the object, its relevant information and time/date of detection is stored.

3.3 Graphical User Interface

A graphical user interface is developed using HTML, CSS, and JavaScript, wherein queries may be performed and certain datasets can be retrieved and plotted via user-defined actions. For example, it would be possible to determine the number of times a certain student brought dangerous objects (e.g. knife/scissors) to the class in a given period of time (e.g. April, May).

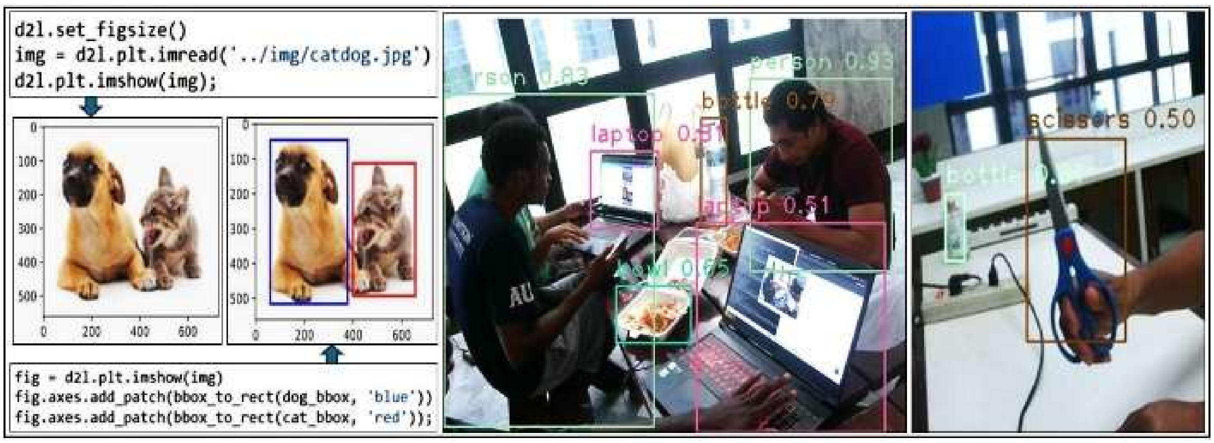
In the overall scheme, Python plays a key role in the development and implementation of the system, serving as the core programming language for handling various tasks, including: object detection and classification, data processing and feature extraction, and integration with databases. The first task is accomplished using machine learning frameworks such as TensorFlow, PyTorch, and OpenCV to detect and classify objects in images. In addition, some previously mentioned state-of-the-art models like YOLO, Faster R-CNN, and SSD are implemented for high-accuracy detection. For Data Processing and Feature Extraction, the captured image provides data preparation facilitation, including image size, generalization, and convenience to improve the performance of the Python model. This allows the use of filtration techniques and changes to increase object recognition and classification accuracy. Finally, python is also necessary for Integration with Databases. In addition to the camera as the main hardware to detect the object(s), a stable network connection (Internet or LAN) is also necessary for database access and real-time monitoring.

4. Implementation and Results

4.1 Object Detection

In order to accomplish the object detection function of our system, **TensorFlow Object Detection API** and **YOLO** were utilized. TensorFlow Object Detection is a powerful API for training and deploying object detection models. It supports various architectures, including previously mentioned SSD, Faster R-CNN, and EfficientDet. In addition, **Ultralytics YOLOv5/YOLOv8**, a fast and efficient PyTorch implementation of YOLO optimized for real-time object detection have been utilized. Finally, our project made use of **OpenCV (cv2.dnn) –** a widely used computer vision library that provides pre-trained object detection models, including SSD, YOLO, and Faster R-CNN. .

Fig. 2 on the following page shows both a generalized object detection and classification example (leftmost part of image), and the actual results of detection and classification from our project. Object detection identifies and locates items within an image using bounding boxes, while classification assigns each detected object to a predefined category based on its features, enabling both spatial and semantic understanding of the scene [12], as seen from the part of the image showing two distinct animals detected and classified. This can be accomplished by writing certain codes in PyTorch. Expanding on this principle and using aforementioned object detection tools, our system has been able to detect distinct objects and also display confidence score of detection as can be seen from Fig. 2. The detected objects were highlighted with bounding boxes and confidence scores, demonstrating the model's accuracy in real-time detection. This visualization showcases the effectiveness of the system in identifying and localizing objects within the scene.



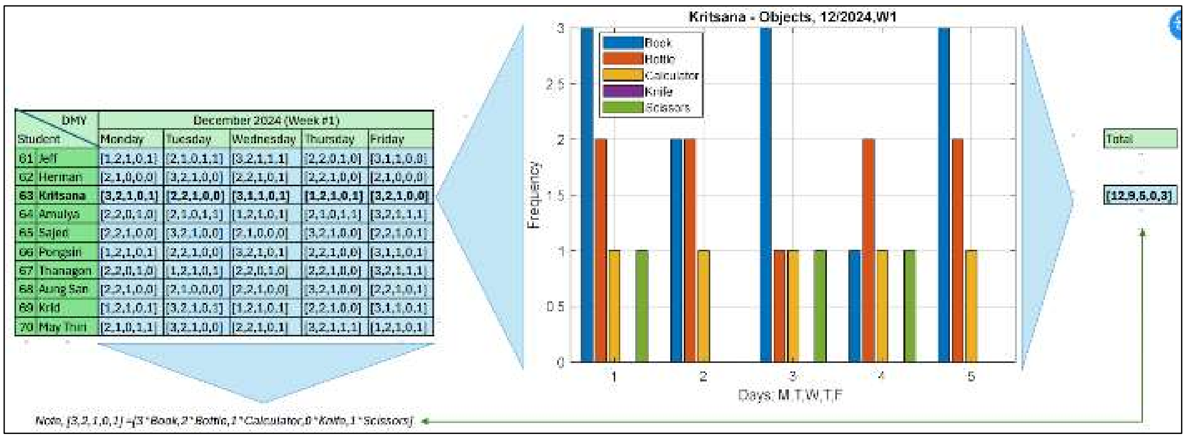
```
d2l.set_figsize()
img = d2l.plt.imread('../img/catdog.jpg')
d2l.plt.imshow(img);
```

```
fig = d2l.plt.imshow(img)
fig.axes.add_patch(bbox_to_rect(dog_bbox, 'blue'))
fig.axes.add_patch(bbox_to_rect(cat_bbox, 'red'));
```

Fig. 2 Object Detection and Classification by the Implemented System

4.2 Database Connectivity

To ensure seamless interaction between the application and the database, our project utilized **Django ORM (Object-Relational Mapper)**, which provides an efficient way to interact with the database using Python, eliminating the need for raw SQL queries. This was done in combination with **SQLite/PostgreSQL** – a relational database used for storing object detection results, user data, and application settings. SQLite is lightweight and suitable for local use, while PostgreSQL offers more advanced features and scalability for larger applications. As key features, the database stores user credentials, preferences, and settings. Within it, a detection records table maintains logs of detected objects, including timestamps, confidence scores, and image paths. A system configuration table holds adjustable parameters such as detection thresholds, alert preferences, and storage settings. Fig. 3 below provides an example of the type of information that may be stored in and retrieved from such a database. It identifies the registered users in the system, and displays information on detected objects carried by them in a certain period of time defined by user query settings.



DMY / Student		December 2024 (Week #1)				
		Monday	Tuesday	Wednesday	Thursday	Friday
61	Jeff	[1,2,1,0,1]	[2,1,0,1,1]	[3,2,1,1,1]	[2,2,0,1,0]	[3,1,1,0,0]
62	Herman	[2,1,0,0,0]	[3,2,1,0,0]	[2,2,1,0,1]	[2,2,1,0,0]	[2,1,0,0,0]
63	**Kritsana**	**[3,2,1,0,1]**	**[2,2,1,0,0]**	**[3,1,1,0,1]**	**[1,2,1,0,1]**	**[3,2,1,0,0]**
64	Amulya	[2,2,0,1,0]	[2,1,0,1,1]	[1,2,1,0,1]	[2,1,0,1,1]	[3,2,1,1,1]
65	Sajed	[2,2,1,0,0]	[3,2,1,0,0]	[2,1,0,0,0]	[3,2,1,0,0]	[2,2,1,0,1]
66	Pongsin	[1,2,1,0,1]	[2,2,1,0,0]	[3,2,1,0,1]	[2,2,1,0,0]	[3,1,1,0,1]
67	Thanagon	[2,2,0,1,0]	[1,2,1,0,1]	[2,2,0,1,0]	[2,2,1,0,0]	[3,2,1,1,1]
68	Aung San	[2,2,1,0,0]	[2,1,0,0,0]	[2,2,1,0,0]	[3,2,1,0,0]	[2,2,1,0,1]
69	Krid	[1,2,1,0,1]	[3,2,1,0,1]	[1,2,1,0,1]	[2,2,1,0,0]	[3,1,1,0,1]
70	May Thiri	[2,1,0,1,1]	[3,2,1,0,0]	[2,2,1,0,1]	[3,2,1,1,1]	[1,2,1,0,1]

Note, [3,2,1,0,1] =[3*Book,2*Bottle,1*Calculator,0*Knife,1*Scissors]

Total
[12,9,5,0,3]

Fig. 3 Information Stored and Displayed Upon Query in System Database

4.3 GUI Development

The main building blocks of this project's GUI implementation consist of HTML, CSS, and JavaScript. Furthermore, additional libraries, such as: Tkinter, PyQt, and Django Templates were also utilized for developing the GUI. Tkinter is a built-in Python library used for creating simple and interactive desktop GUIs. PyQt is powerful Python library for developing cross-platform desktop GUIs with advanced widgets and layouts. Django Templates are used for rendering dynamic web-based user interfaces within the Django framework, allowing for the creation of dynamic and responsive web pages.

The GUI was designed with the intention to provide an intuitive and user-friendly experience for interacting with the object detection system. The design considerations included responsiveness, ease of use, and real-time visualization of detection results. It features a live camera feed, displaying the real-time video stream with detected objects highlighted using bounding boxes, the option to upload and analyze, allowing users to upload images or videos for object detection analysis, a results display showing detected objects, confidence scores, and timestamps in an easy-to-read format, and settings and configurations – allowing users to adjust model parameters, detection thresholds, and storage settings.

Additionally, real-time updates and alerts were integrated into the system. Live detection feedback ensures that the detected objects are instantly displayed on the screen with their classification and detection confidence levels. Visual alerts in the form of notifications are triggered when specific objects of interest are detected. Then detected object information is automatically stored in the database for future reference and analysis. An example of this alert is shown in Fig. 4. The current alert information includes objects detected, location of detection, time-stamp, and alert recipient.

As mentioned in the earlier Section 3, Python played a key role in the implementation of our overall system at different stages. This was accomplished by importing and utilizing various libraries offered by Python. Shown below is a Python code snippet, used in the project to import some of the required libraries:

*“**import** os*

***import** uuid*

***from** django.db **import** models*

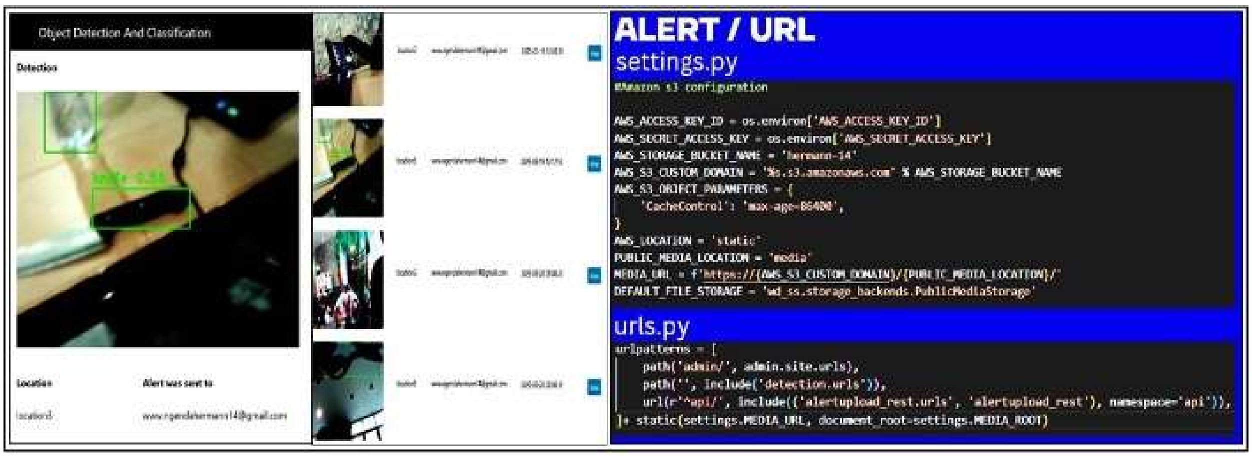
***from** django.conf **import** settings*

***from** django.db.models.signals **import** post_save*

***from** django.dispatch **import** receiver*

***from** rest_framework.authtoken. models **import** Token*

***from** django.contrib.auth.models **import** User*



```
#Amazon s3 configuration

AWS_ACCESS_KEY_ID = os.environ['AWS_ACCESS_KEY_ID']
AWS_SECRET_ACCESS_KEY = os.environ['AWS_SECRET_ACCESS_KEY']
AWS_STORAGE_BUCKET_NAME = 'hermann-14'
AWS_S3_CUSTOM_DOMAIN = '%s.s3.amazonaws.com' % AWS_STORAGE_BUCKET_NAME
AWS_S3_OBJECT_PARAMETERS = {
    'CacheControl': 'max-age=86400',
}
AWS_LOCATION = 'static'
PUBLIC_MEDIA_LOCATION = 'media'
MEDIA_URL = f'https://{AWS_S3_CUSTOM_DOMAIN}/{PUBLIC_MEDIA_LOCATION}/'
DEFAULT_FILE_STORAGE = 'wd_ss.storage_backends.PublicMediaStorage'
```

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('detection.urls')),
    url(r'^api/', include(('alertupload_rest.urls', 'alertupload_rest'), namespace='api')),
]+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Fig. 4 System-Generated Alert

5. Educational Impact and Curriculum Integration

This AI-assisted object detection project serves as a valuable educational platform by bridging theory and practice in engineering education. As a senior-year undergraduate endeavor, it fosters project-based learning where students apply interdisciplinary knowledge in machine learning, computer vision, software engineering, and database management using industry-standard tools like YOLO, TensorFlow, OpenCV, Django, and SQLite. The project also introduces front-end development and user-centered design through GUI and alert system implementation, broadening students’ skillsets beyond core algorithms. Faculty mentorship ensures students grasp ethical, security, and societal implications, promoting responsible engineering.

Embedding such AI-focused, student-led projects into final-year design courses or labs aligns with modern curriculum goals by providing hands-on experience across the software lifecycle while addressing real-world issues like campus safety. This approach prepares graduates to tackle the technical and social challenges of AI-driven systems effectively.

6. Discussion and Conclusion

Recent advancements in AI and computer vision have enabled diverse applications, exemplified by the object detection system developed by senior undergraduate Computer Engineering students in this project. The work integrates object detection, database management, and GUI development using Python libraries (YOLO, OpenCV, TensorFlow), MySQL/Django/SQLite, and web technologies to create an effective security solution tailored for educational environments.

By reducing reliance on human monitoring, the system improves efficiency, minimizes fatigue-related errors, and offers scalability through integration with existing CCTV networks. However, challenges such as false positives and negatives, privacy concerns, financial investment, and dependence on the diversity of training data remain significant and warrant careful consideration. Ethical issues surrounding privacy, psychological impact, and data security must be addressed responsibly to ensure trust and proper system use.

Beyond its technical achievements, this project demonstrates the educational value of AI-focused, student-led initiatives that provide practical, interdisciplinary learning experiences across the full software development lifecycle. Faculty mentorship ensures students develop not only technical skills but also an understanding of ethical and societal implications. Embedding such projects into engineering curricula prepares graduates to meet the evolving technical and social challenges posed by AI applications, reflecting the future direction of engineering education.

During this student-implemented project, feedback gathered from project advisors (faculty members) offered a foundation for ongoing development, serving as a roadmap for future student projects within the department. The feedback comments were categorized into the following areas: detected objects' images, database system, information display and query, and future improvements. It is summarized in Table 1 shown on the right-hand column of this page.

Table 1 Feedback from Testing; and Future Considerations.

Category	Details of Feedback	Proposed Improvements
Detected Objects' Images	Good video performance; include images of detected objects.	Extract and include sample images of detected objects.
Database System	Ensure database structure connects recognized objects to users (i.e. make it relational), with timestamps and MySQL verification.	- Confirm and document that Django is used for storage. - Provide a database schema linking objects to users with timestamps. - Ensure logs include object type, user ID, date, time, and location.
Information Display and Query	Improve visual representation and querying for specific object detections (e.g., total knives detected per week).	- Enhance GUI with search and filter options. - Implement summary statistics and visualization tools (e.g., bar charts, tables). - Demonstrate query examples for retrieving reports based on object type, user history, or time frame.
Future Improvements	- Reduce false positives/negatives, especially for small/occluded objects. - Training dataset should be larger for better reliability. - Database queries could have been faster - Encryption and access control required for data security.	- Fine-tune the model to improve detection accuracy. - Expand the dataset for better performance in diverse environments. - Optimize database queries for speed and efficiency. - Implement encryption and access controls to protect stored data.

Acknowledgement

The authors cordially thank their respective faculties and departments for their support and encouragement in the publication of this work.

References

[1] Ekpoh, U. I., Edet, A. O. and Ukpong, N. N. (2020). Security challenges in universities: Implications for safe school environment. *Journal of Educational and Social Research*, vol. 10, no. 6, pp. 112–112, Nov. 2020, doi: 10.36941/jesr-2020-0113.

[2] Soji, E. S. et al (2024). AI-driven computer vision for intelligent home automation and surveillance systems. In *Explainable AI Applications for Human Behavior Analysis*, P. Paramasivan *et al*., Eds. Hershey, PA, USA: IGI Global, 2024, pp. 242–257. doi: 10.4018/979-8-3693-1355-8.ch015.

[3] GeeksforGeeks, "Real-time systems," *GeeksforGeeks*, [Online]. Available: https://www.geeksforgeeks.org/real-time-systems/. [Accessed: Feb. 10, 2025].

[4] Malisiewicz, T., Gupta A. and Efros, A. A. (2011). Ensemble of exemplar-SVMs for object detection and beyond. *2011 International Conference on Computer Vision (ICCV)*, Barcelona, Spain, 2011, pp. 89–96, doi: 10.1109/ICCV.2011.6126229.

[5] Dalal, N. and Triggs, B. (2025). Histograms of oriented gradients for human detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, San Diego, CA, USA, 2005, pp. 886–893, vol. 1, doi: 10.1109/CVPR.2005.177

[6] Bansal, M., Kumar, M. and Kumar, M. (2021). 2D object recognition: a comparative analysis of SIFT, SURF and ORB feature descriptors. *Multimedia Tools and Applications*, vol. 80, no. 12, pp. 18839–18857, May 2021, doi: 10.1007/s11042-021-10646-0.

[7] Girshick, R. Fast R-CNN. In Proc. *2015 IEEE Int. Conf. Comput. Vis. (ICCV)*, Santiago, Chile, 2015, pp. 1440–1448, doi: 10.1109/ICCV.2015.169.

[8] GeeksforGeeks, "R-CNN vs Fast R-CNN vs Faster R-CNN | ML," *GeeksforGeeks*, [Online]. Available: https://www.geeksforgeeks.org/r-cnn-vs-fast-r-cnn-vs-faster-r-cnn-ml/. [Accessed: Feb. 10, 2025].

[9] Redmon, J. and Farhadi A. (2017). YOLO9000: Better, Faster, Stronger. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017, pp. 6517-6525, doi: 10.1109/CVPR.2017.690.

[10] Dosovitskiy, A. et al. (2021). An image is worth 16x16 words: transformers for image recognition at scale. In *Proc. Int. Conf. Learn. Representations (ICLR)*, 2021.

[11] Tanenbaum, A. S. and Van Steen, M. (2007). *Distributed Systems: Principles and Paradigms*, 2nd ed. Upper Saddle River, NJ, USA: Pearson Prentice Hall, 2007.

[12] Zhang, A., Lipton, Z. C., Li, M. and Smola, A. J.(2020). *Dive into Deep Learning*, 1st ed. d2l.ai, 2020.