An integrated intelligent surveillance system for Industrial areas

Francesco Camastra, Angelo Ciaramella, Angelo Casolaro, Pasquale De Trino, Alessio Ferone, Giovanni Hauber, Gennaro Iannuzzo, Vincenzo Mariano Scarrica, Antonio Junior Spoleto, Antonino Staiano* and Maria Concetta Vitale

Department of Science and Technology, Parthenope University of Naples, Centro Direzionale Isola C4, Naples, 80143, Italy

Abstract

This paper presents the design and implementation phases of a software prototype developed by the University of Parthenope for the SE4I project (*Smart Energy Efficiency & Environment for Industry*), funded by the "Progetti di ricerca industriale e lo Sviluppo sperimentale" (PNR 2015-2020). The prototype leverages advanced computer vision techniques based on deep learning architectures to address industrial security and monitoring needs. Specifically, the prototype tackles three key functionalities, (1) personnel and vehicle identification: The system recognizes authorized personnel and vehicle license plates within video streams captured in restricted industrial areas; (2) anomaly detection: The software can detect various anomalies in video feeds, including falls of personnel in monitored zones and unattended objects left in unauthorized areas; (3) smart parking management: The prototype identifies vacant parking spaces within camera-monitored zones, enabling efficient parking management. These functionalities are integrated into the software prototype, and its performance has been thoroughly evaluated.

Keywords

Plate Detection, Face Detection, Fall Detection, Parking Detection

1. Introduction

The SE4I project aims to improve safety within a designated industrial area by implementing a real-time video monitoring system. This system uses strategically placed smart poles equipped with RGB cameras to capture video streams. The project focuses on three key functionalities (see Fig. 1): (a) authorized access control: The system will recognize individuals and vehicle license plates. This ensures that only authorized personnel and vehicles can enter the area, likely through controlled access points with barriers. Upon arrival, an employee's car triggers the system. The camera mounted on the smart pole captures the RGB video stream of the scene, using AI to identify the license plate and the driver's face. Access is then granted only after successful recognition. The combined recognition of license plate and driver verifies that the vehicle and driver are authorized. If recognition is successful, access is allowed; if not, access is denied; (b) anomaly detection: This use case focuses on detecting abnormal behavior or events in the video streams captured by the pole-mounted cameras. These anomalies can range from environmental violations, such as illegal dumping of waste to personnel safety issues, e.g., persons' falls, and unattended objects left in restricted areas. An RGB video stream from a smart pole camera continuously feeds the scene, which may include both facility personnel and outsiders, to a hardware component equipped with AI modules. The intelligent module analyzes the video to identify unusual elements. Upon detection, an alert is sent to a central control station that specifies the type and location of the event. This allows immediate assistance to personnel experiencing medical emergencies or accidents and swift intervention in case of suspicious activity; (c) smart parking management: This use case addresses the detection and management of parking availability within the vast industrial area. Due to its size, automated and intelligent parking lot monitoring is crucial. This system will inform users about free parking spaces as they approach designated parking spaces. In the context of performing learning tasks from video streams in surveillance and security appli-

Ital-IA 2024: 4th National Conference on Artificial Intelligence, organized by CINI, May 29-30, 2024, Naples, Italy

*Corresponding author.

francesco.camastra@uniparthenope.it (F. Camastra); angelo.ciaramella@uniparthenope.it (A. Ciaramella); angelo.casolaro001@studenti.uniparthenope.it (A. Casolaro); pasquale.detrino001@studenti.uniparthenope.it (P. D. Trino); alessio.ferone@uniparthenope.it (A. Ferone); giovanni.hauber@studenti.uniparthenope.it (G. Hauber); gennaro.iannuzzo001@studenti.uniparthenope.it (G. Iannuzzo); vincenzomariano.scarrica001@studenti.uniparthenope.it (V. M. Scarrica); antoniojunior.spoleto001@studenti.uniparthenope.it (A. J. Spoleto); antonino.staiano@uniparthenope.it (A. Staiano); mariaconcetta.vitale001@studenti.uniparthenope.it (M. C. Vitale)

0000-0003-4439-7583 (F. Camastra); 0000-0001-5592-7995 (A. Ciaramella); 0000-0002-7577-6765 (A. Casolaro); 0009-0003-0680-4501 (P. D. Trino); 0000-0002-4883-0164 (A. Ferone); 0009-0007-0137-3182 (G. Hauber); 0009-0003-5962-8302 (G. Iannuzzo); 0009-0008-4640-2693 (V. M. Scarrica); 0009-0007-4037-7821 (A. J. Spoleto); 0000-0002-4708-5860 (A. Staiano); 0000-0002-5538-9952 (M. C. Vitale)

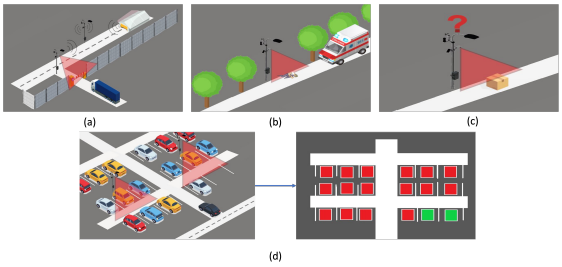

Figure 1: Project tasks: (a) people/vehicles recognition; (b) fall detection; (c) anomaly detection; (d) parking lot handling.

cations, the state of the art is represented by computer vision techniques based on the use of deep learning techniques [1, 2].In the subsequent sections, we will discuss the proposed solutions for each of the aforementioned tasks.

2. Plate Detection

The objective is to recognize vehicles and their license plates from a surveillance video feed and retrieve the associated alphanumeric sequence. This sequence is subsequently utilized for additional vehicle recognition within the system.

2.1. Challenges

Automatic license plate recognition faces several hurdles, specifically, (a) Variable Lighting: Extreme brightness, low light, and shadows can significantly reduce plate visibility. The systems address this with techniques like adaptive thresholding and contrast enhancement; (b) Car Position: Vehicles approach cameras at various angles and distances. Sophisticated algorithms are required for accurate plate localization and perspective correction to account for these variations; (c) Occlusions: Objects like bumpers, dirt, or even other vehicles can partially or fully obscure the plate. Robust object detection and diverse training data are crucial to overcome these occlusions; (d) Font Diversity: License plate formats and fonts differ significantly across countries and even regions. Training models on a wide variety of datasets is essential for generalization across different plate styles.

2.2. Methods

The proposed approach consists of three key steps: vehicle detection, license plate (LP) detection, and optical character recognition (OCR), as shown in Figure 2. In the first step, the system detects vehicles in the scene using a dedicated module. Within each detected vehicle region, the Warped Planar Object Detection Network (WPOD-NET) [3] is used to search for license plates. The WPOD-NET performs affine transformations to rectify the LP area to resemble a frontal view. These detections are then passed to an OCR network for accurate character recognition and extraction. To balance computation time and performance, we chose YoloV4 [4]. For the classification problem, we treated the network as a closed system, consolidating the outputs specifically related to vehicles such as cars, buses, and motorcycles, while ignoring outputs related to other classes. The design of

Figure 2: The proposed pipeline at work.

WPOD-NET, which is responsible for warping the license plate into a rectangular shape, was influenced by insights from YOLO, SSD [5], and Spatial Transformer Networks (STN) [6]. Finally, in our OCR module, we used Tesseract [7], a fine-tuned optical character recognition engine trained on our license plate character dataset. Tesseract's advantage over a simple CNN lies in its recurrent neural network (RNN) architecture [8], which takes into account the sequential nature of the characters on a license plate. This allows for accurate recognition as the RNN captures the contextual dependencies between characters. Tesseract's extensive training on diverse datasets makes it robust, handling different font styles, sizes, and noise levels commonly found in license plate images.

2.3. Execution

The modules are designed for real-time execution in an embedded system environment, given the strict time constraints imposed by vehicle identification. Fig. 3 illustrates the time required for the execution steps.

PLATE DETECTION		
Step	**Half precision***	**Full precision***
Car Detection	25 fps	20 fps
Plate Detection	10 fps	6 fps
OCR	30+ fps	30+ fps
Executed on the Embedded Nvidia **Jetson Xavier AGX**. "**Half**" and "**Full precision**" refer to the modes in which **numerical calculations** are performed for these modules, directly **impacting both the accuracy of the result and the execution speed.**		

Figure 3: Execution time of the pipeline.

3. Face Recognition

Our research team has developed a framework for secure access control in the industrial area as a smart city environment. The framework leverages surveillance cameras positioned at entry points to industrial areas. It aims to match detected faces with the license plates of associated vehicles. This ensures that the driver corresponds to the registered vehicle owner, improving access control security.

3.1. Challenges

The framework faces different issues, particularly with on-board processing. In addition, reflective surfaces and occlusions caused by sunlight on vehicle windshields can hinder facial recognition.

3.2. Methods

The face recognition process is divided into two main steps, namely, face localization and face classification. In the localization step, faces are accurately localized within an image by a Multi-task Cascaded Convolutional Networks (MTCNN) algorithm [9]. Its unique cascade structure consists of three stages: Proposal Network (P-Net), Refinement Network (R-Net), and Output Network (O-Net). By simultaneously performing multiple tasks such as face detection, bounding box regression, and face landmark localization, MTCNN ensures a thorough and accurate face identification. In particular, it excels at detecting faces at different scales and orientations while maintaining impressive computational efficiency, making it ideal for real-time applications.

For face classification, an extremely effective approach combines two different algorithms. The first is a face alignment with an ensemble of regression trees [10]. Using an ensemble of regression trees, the algorithm predicts the positions of facial landmarks directly from image data, bypassing traditional optimization methods. The second uses FaceNet[11], which efficiently maps a face into a continuous embedding space, i.e., converting it into a 128-feature embedding vector. This vector is then matched to a face in the database using a one-shot approach (see Fig. 4 as an example of qualitative result on a test image.)

4. Anomaly Detection

Anomaly detection involves the identification of unusual items, events, or observations that are significantly different from the norm or expected behavior, or that indicate unusual conditions. The activities conducted in the SE4I project focused on identifying waste dumping where access is prohibited, and fall detection.

Figure 4: Visual results on a customized test.

4.1. Challenges

Anomaly recognition in video streams presents a significant challenge: identifying rare and short-lived events that deviate from the norm. These anomalies often occur for just a few seconds, making them difficult for humans to detect and nearly impossible to capture in a single, universal model. The vast number of possible anomaly types, locations, and contexts makes defining a comprehensive model impractical. It would require an enormous amount of data and manual effort. A more effective approach is to train models that can differentiate between normal and abnormal activity, regardless of the specific anomaly type. This approach leverages the fact that normal behavior typically occurs far more frequently than anomalies.

4.2. Methods

We have used a reconstruction-based method, where a model is trained to learn the normal patterns of the data, to be able to reconstruct them when new frames are presented. The model is designed to extract the spatiotemporal structure of the video stream to accurately learn the pattern of normality for a scene without anomalies [12](without the abnormal situation highlighted in red in Figure 5). During testing, the model provides information about the most anomalous areas of the video by comparing input frames with reconstructed frames. A relatively low score indicates a normal scene, while a high score indicates the presence of anomalies. The goal is to create an end-to-end model capable of learning the spatio-temporal patterns of the analyzed data and predicting when an event is anomalous compared to the learned normality. The model used for this unsupervised analysis is called CLSTM-AE, which stands for Long-Short Term Memory Convolutional-Transpose Convolutional Autoencoder. The architecture is a special type of autoencoder [13]. This approach enables real-time anomaly detection by identifying events that

Figure 5: Anomaly detection in a parking lot.

deviate significantly from the learned representation of normal video data. The trained model compares the reconstructed video frames with the original input. For normal events, the reconstructed frames closely resemble the originals, with minimal differences in pixel values. However, when anomalies occur, the network's reconstruction becomes less accurate. This is reflected in blurry or distorted frames compared to the originals. By analyzing this reconstruction error (the difference between original and reconstructed frames), one can identify anomalies in real-time.

4.3. Fall Detection

Falls, especially outdoors where help might be delayed, can lead to serious injuries. Traditional fall detection systems often rely on wearable sensors, which can be inconvenient or impractical. The SE4I project addresses this challenge with a camera-based fall detection system using an LSTM Autoencoder. This system leverages anomaly detection techniques within a computer vision framework. It essentially learns what "normal" movement looks like and identifies deviations from this norm as potential falls. This approach offers several advantages, specifically, no need for wearable Sensors, only camera-based detection working with existing surveillance infrastructure, and the use of real-time alerts, enabling faster response times.

4.3.1. Challenges

Traditional fall detection systems typically rely on wearable sensors or specialized depth cameras. These methods can be intrusive to users and costly to deploy on a large scale. On the other hand, relying solely on human observation through video footage is an option. However, this approach is labor-intensive and requires continuous monitoring.

4.3.2. Methods

Our method addresses the previous issues by enabling fall detection with a simpler setup, a standard RGB camera, eliminating the need for specialized equipment, and AI-powered detection that uses a single AI module running on a GPU to analyze the video stream for instances of falls. This approach eliminates the need for wearable devices and reduces the reliance on human intervention. Detection is to be performed in pedestrian areas and parks, and so a dataset was created to fit this particular environment and to train the model on data representing the final context. The training dataset is illustrative of all the normal poses that people take while walking in places like pedestrian areas and parks. The scenes were therefore captured with a fixed camera about 3 meters above the ground, pointing across an open space and covering walking, standing poses, and running events in all directions and with or without obstacles/occlusions. The idea of training a model with only "normal events" is important because, in nature, abnormal events (falls) are very rare and therefore expensive to acquire. The data preprocessing pipeline involves using the openpose framework [14] to extract skeleton key points from each frame. From each skeleton, irrelevant keypoints are removed as they are considered noisy and skeletons with significant missing keypoints are filtered out. Keypoint coordinates are then normalized using min/max normalization and discretized into coarser bins to provide numerical stability for the training phase. Finally, the data are shaped into time windows using sequences of 75 skeletal frames. Such windows form the basic unit on which the AI model operates. Since the video stream is supposed to be captured at 25 FPS, working with 75 frame windows means analyzing human behavior over 3-second actions. An overlap of 25 frames between consecutive windows is also included to maintain continuity between the windows themselves. The model is based on a LSTM autoencoder [15], [16]. The execution time when running on a consumer GPU allows for real-time performance (see Fig. 6). Once the model has learned normal

Step	Execution time	Scope
Openpose	0.0294 sec	One frame
Preprocessing	0.1586 sec	One window
Inference	0.2354 sec	One window

Figure 6: Pipeline components execution times.

human behavior patterns, it can be used to reconstruct time windows. Reconstruction and input data are then compared; if the reconstruction error exceeds a certain

threshold and deviates significantly from normal data, the input is intuitively flagged as a fall event. Overall, the results highlight the effectiveness of using learned temporal skeletal patterns for robust anomaly detection in the context of outdoor fall detection.

5. Parking Detection

Parking detection for SE4I requires the development of an automatic system that searches for *free parking space* in one of the parking areas within the industrial area and provides information to drivers who have requested a parking space. The *Parking Guide and Information* (PGI) system [17] has been adopted as a solution for the parking detection task using a monitoring system. The proposed PGI system consists of two main parts. The former is based on *deep learning instance segmentation model* to detect all available free spaces in a parking lot. The latter is a *client-server* architecture that automatically guides drivers to the closest parking lot with the highest number of available spaces.

5.1. Challenges

Parking lot detection systems using video surveillance face several difficulties: (a) the impact of weather, e.g., low visibility caused by fog, rain, and snow can significantly decrease the accuracy of these systems, or harsh weather conditions can obscure parking lot boundaries in the video feed; (b) diverse parking lot data, i.e., training robust parking detection models requires a large dataset with a wide variety of scenarios, including variations in parking space layouts, weather conditions, camera angles, obstructions, parking lot types (e.g., open-air, multi-story), and lighting conditions (day/night); (c) real-time processing, that is, for practical applications, the system needs to operate in real-time, this necessitates developing a light parking detection model that can run efficiently on available hardware.

5.2. Methods

This work conceived a model for parking lot detection using an instance segmentation approach. Yolact++ [18], which is an extension of Yolact [19], was trained with successful results on a novel dataset appropriately designed for this task. The dataset consists of 1395 images and 23600 manually annotated parking lots, and it was built by using a web-scarping approach. The images, taken from public access cameras, were selected to represent a variety of conditions, i.e. *weather and lighting conditions*, features, i.e, *different camera angles*, *occlusions*, *shadows*, *presence of people or animals*, *camera heights*, *satellite imagery in 2D and 3D*, different types of lines and colors, and different backgrounds.

Parking lot detection and car detection are performed simultaneously to classify occupied or free parking lots on the basis of the IoU between parking lot and car masks detected by the Yolact++ module, respectively. For IoU values greater than an IoU threshold, the system classifies parking lots as *busy lots*, as *free lots*, otherwise.
The Yolact++ architecture is based on the RetinaNet architecture [20], using pre-trained ResNet-101 stages. In addition, Yolact++ introduces three improvements over the base model: *Fast Mask Re-Scoring Network Stage*, *Deformable Convolutions with Intervals*, and a *Optimized Prediction Head*. The selection of the Yolact++ architecture for the parking lot detection problem was motivated by the runtime requirements and the accuracy achieved by this instance segmentation model.
A client-server system called PGI has been developed. The clients include the *drivers*, *administrators*, and *machine learning* systems. Drivers can search for parking lots, while administrators can add, remove, and monitor parking lots.System operations are performed on the server side, which is built using PHP and MySQL for database storage. Clients connect to the server through a server interface using a Java Android app. The app provides various functionalities, such as guiding drivers to the nearest parking lot with available spaces using GPS, and monitoring areas using the Google StreetView API. The system presents favorable results with low loss values and acceptable mAP for both the box and the mask, determined using a 0.5 IoU threshold (see Table 1 and Fig. 7).

Metrics	Average Values
Box Localization Loss	2.027
Class Confidence Loss	1.604
Mask Loss	3.185
Semantic Segmentation Loss	0.125
I Loss	0.116
Total Loss	7.058
mAP@0.50 Box	80.5
mAP@0.50 Mask	76.62

Table 1
Results on Yolact++ after fine-tuning and testing on the custom dataset.

6. Integration and Infrastructure

The five intelligent modules of the SE4I project, *plate detection*, *face detection*, *anomaly detection*, *fall detection*, and *parking detection*, are part of a larger system powered by a peer-to-peer network of **NVIDIA Jetson Xavier** devices mounted on multifunctional light poles. This setup ensures efficient and real-time processing of the data collected by the surveillance cameras, as the computation is performed in the field and each device shares

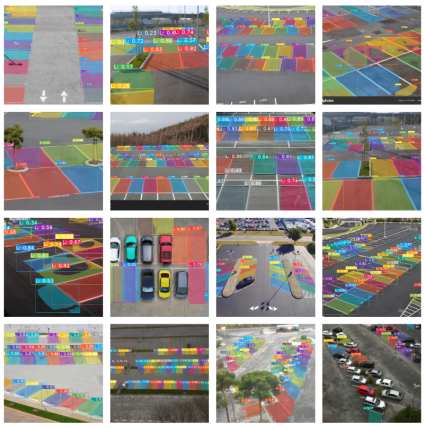

Figure 7: Visual results on test images. Masks are applied to the lots, each with a different color to better distinguish each instance. The associated probability score is printed on each mask.

data and JSON output with devices on other poles using a *ZMQ* publisher/subscriber pattern. Therefore, a dedicated module manages the surveillance camera stream and associated metadata, such as *brightness*, *frame rate*, *contrast*, etc. All modules are containerized as *Docker* solutions, allowing for flexible portability, easy installation, resilience, and scalable performance. The whole system is based on Python and C++ programming languages and PyTorch, OpenCV, OpenPose, Onvif libraries are used. This infrastructure guarantees the real-time requirements and the privacy of the video-monitored areas.

Acknowledgments

This research was conducted as part of the *Smart Energy Efficiency & Environment for Industry (SE4I)* project, CUP *I66G*18000230005, funded by "Progetti di ricerca industriale e lo Sviluppo sperimentale nelle 12 aree di specializzazione individuate nel PNR 2015-2020, di cui al D.D. del 13 luglio 2017 n. 1735".

References

[1] A. Ferone, A. Maratea, Adaptive quick reduct for feature drift detection, Algorithms 14 (2021).

[2] A. Maratea, A. Ferone, Deep neural networks and explainable machine learning, in: WILF 2018, volume 11291 LNAI, 2019, p. 253 – 256.

[3] S. Montazzolli, C. Jung, License plate detection and recognition in unconstrained scenarios, in: ECCV 2018, Springer Intl. Pub., 2018, pp. 593–609.

[4] A. Bochkovskiy, C. Wang, H. M. Liao, Yolov4: Optimal speed and accuracy of object detection, CoRR abs/2004.10934 (2020). arXiv:2004.10934.

[5] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg, Ssd: Single shot multibox detector, in: ECCV 2016, Springer Intl. Pub., 2016, pp. 21–37.

[6] M. Jaderberg, K. Simonyan, A. Zisserman, K. Kavukcuoglu, Spatial transformer networks, CoRR abs/1506.02025 (2015).

[7] R. Smith, An overview of the tesseract ocr engine, in: ICDAR 2007, volume 2, 2007, pp. 629–633.

[8] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (1997) 1735–80.

[9] K. Zhang, Z. Zhang, Z. Li, Y. Qiao, Joint face detection and alignment using multitask cascaded convolutional networks, IEEE Signal Processing Letters 23 (2016) 1499–1503.

[10] V. Kazemi, J. Sullivan, One millisecond face alignment with an ensemble of regression trees, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1867–1874.

[11] F. Schroff, D. Kalenichenko, J. Philbin, Facenet: A unified embedding for face recognition and clustering, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 815–823.

[12] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury, L. S. Davis, Learning temporal regularity in video sequences, 2016. arXiv:1604.04574.

[13] Y. S. Chong, Y. H. Tay, Abnormal event detection in videos using spatiotemporal autoencoder, 2017. arXiv:1701.01546.

[14] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, Y. Sheikh, Openpose: Realtime multi-person 2d pose estimation using part affinity fields, 2019. doi:10.1109/TPAMI.2019.2929257. arXiv:1812.08008.

[15] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997).

[16] M. A. Kramer, Nonlinear principal component analysis using autoassociative neural networks, Aiche Journal 37 (1991) 233–243.

[17] D. Acharya, W. Yan, K. Khoshelham, Real-time image-based parking occupancy detection using deep learning., Research@ Locate 4 (2018) 33–40.

[18] C. Zhou, Yolact++ Better Real-Time Instance Segmentation, University of California, Davis, 2020.

[19] D. Bolya, C. Zhou, F. Xiao, Y. J. Lee, Yolact: Real-time instance segmentation, in: Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 9157–9166.

[20] T. Lin, P. Goyal, R. B. Girshick, K. He, P. Dollár, Focal loss for dense object detection, CoRR abs/1708.02002 (2017). URL: http://arxiv.org/abs/1708.02002. arXiv:1708.02002.