This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/ACIT62805.2024.10877023, 2024 25th International Arab Conference on Information Technology (ACIT)

A Computer Vision Model to Support Individuals with Disabilities Within University Campuses

Allan Costa Nascimento dos Santos*,†,‡,*, Karina de Paula$^{\star,\mp}$, Marcos T. L. Vidal*, João M. M. da Silva*, Cledson de Sousa*,*, Leandro A. F. Fernandes†, Tiago B. de Castro*,*, Marcos Bedo†,Troy C. Kohwalter†, Carlos A. M. Bastos*,*, Flavio L. Seixas†,*, Natalia C. Fernandes*, Débora C. Muchaluat-Saade†,*, Gheorghita Ghinea‡
**GTECCOM*, **MidiaCom and* $^{\mp}$*INCT-INEAC-UFF Labs*
†*Institute of Computing – IC/Universidade Federal Fluminense – Niterói, RJ – Brazil*
‡*Department of Computer Science/Brunel University London – London, United Kingdom*
{allans, kp,mvon, jmarcos, cledsons, tiago_bornia, camalcherbastos, nataliacf}@id.uff.br,
{laffernandes, marcosbedo, troy, fseixas, debora}@ic.uff.br, {Allan.Santos, george.ghinea}@brunel.ac.uk

***Abstract*—This study introduces MCOF, a multi-camera, computer vision-based system designed to assist visually impaired individuals with mobility on university campuses. The system operates locally and achieves the following performance metrics: *(i)* detecting body points within $1.2 \cdot 10^{-2}$ seconds, *(ii)* identifying people, objects, and animals in $2.4 \cdot 10^{-2}$ seconds, and *(iii)* detecting movements in $5.1 \cdot 10^{-2}$ seconds. These results were obtained using a GTX 1,660 GPU with up to 6 cameras (or a 6,112MB stream) running concurrently. According to the MCOF architecture, events trigger tickets that are sent to an external information system, which can then implement its own safety and personnel protocols. Additionally, MCOF includes modules to handle electrical and network failures and features an obstruction detection routine for the cameras.**

***Index Terms*—Fall Detection, Computer Vision, Healthcare, Human Activity Recognization, Campus Safety, Proactive Security**

I. Introduction

Vision impairment affects approximately 1.1 billion people worldwide, according to Vision Atlas [2]. This condition, which can lead to partial or total vision loss [3], has a profound impact on both quality of life and self-esteem. Tasks that are routine for sighted individuals, such as safely navigating indoor and outdoor spaces, become challenging for those with vision loss. Additionally, advancements in technology have contributed to an aging population, with the proportion of individuals aged 65 and above compared to those aged 15–64 expected to reach 54 percent by 2050 in the EU [4].

Notoriously, visually impaired individuals often face a significant risk of falls and injuries due to difficulties in identifying hazards while moving [2]. Therefore, it is crucial to alert them to potential dangers in both familiar and unfamiliar environments. As a result, obstacle detection has become a key aspect of mobility assistance for the visually impaired. Utilizing computer vision for obstacle detection can be instrumental in improving accessibility for young individuals with disabilities, especially on college campuses.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, CAPES Print, CNPq, FAPERJ, UFF and FINEP. Acknowledgments to the house Bem Estar - Israeli Home for Elderly Care [1] for supporting the research.

Obstacle detection enables individuals with disabilities to navigate physical barriers such as steps, floor obstacles, or misplaced objects. This capability enhances their academic experience and promotes smoother integration into the university community. By employing a multi-camera system integrated with computer vision, individuals with disabilities can achieve greater independence while moving through a university campus, reducing their reliance on external assistance. The ability to detect obstacles simplifies the mobility of students with disabilities, making it easier for them to access classrooms, libraries, laboratories, and other campus locations, thereby improving their overall efficiency.

In this study, we introduce MCOF, a multi-camera, computer vision-based system designed to enhance mobility for visually impaired individuals on university campuses by detecting falls and obstructions. The quicker the model identifies a fall or obstruction, the faster security staff can respond to assist or remove the obstacle. The system utilizes campus cameras to detect objects in the environment. With numerous cameras covering pathways and corridors, the model automatically identifies objects left in these areas. If an object remains in the path for an extended period, the system generates an alert and sends it to the messaging system through a ticketing mechanism, operating in client-server mode.

The ticket message includes key details such as name, content, department, priority, and even georeferencing information. MCOF operates as a real-time video streaming system with image processing capabilities using Python. The client (browser) sends an HTTP request to MCOF, which then creates a thread to stream the camera feed to the client. MCOF captures video frames from the camera, performs image preprocessing, and applies algorithms for object detection, person detection, and body points detection.

The remainder of this paper is organized as follows. Section II discusses background concepts and related work. Section III details the proposed model, and Section IV presents the experimental evaluation. Finally, Section V offers the conclusions and outlines future work.

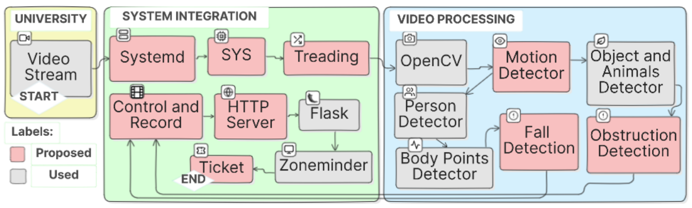

Fig. 1. Dataflow in `MCOF` begins with the video stream originating from a university camera. The `MCOF` Systemd, SYS, and Threading modules receive this data to initiate system integration. Next, the `MCOF` OpenCV, Motion Detector, Person Detector, Body Points Detector, and Fall Detection modules work together to identify falls. Simultaneously, the `MCOF` Object and Animals Detector and Obstruction Detector modules detect obstructions. Finally, the `MCOF` Control and Record, HTTP Server, Flask, Zoneminder, and Ticket modules are used to record incidents and trigger alarms for both obstructions and falls.

II. Related Work

Nikhil et al. (2021) [5] proposed a system deployed on a university campus to ensure the safety and hygiene of individuals. This system includes three main modules: Crowd Counting, Social Distancing, and Mask Detection, and it continuously monitors activities using Computer Vision technology. Ritharson et al. (2022) [6] developed a preventive method for detecting individuals wearing masks or carrying weapons, whether lethal or non-lethal, utilizing the YOLOv3 algorithm for detection. Ashraf et al. (2022) [7] created a model for detecting pistols with sufficient speed for alarm-based surveillance systems, implementing YOLOv5s, which demonstrates effective results and speed.

Guler et al. (2016) [8] described a system based on background subtraction that includes motion detection, camera sabotage detection (covering issues such as moved, out-of-focus, and covered cameras), abandoned object detection, and object-tracking algorithms. Similarly, Vijetha et al. (2024) [2] presented an obstacle detection system designed to aid visually impaired individuals in navigating using smartphones. Their approach combines semantic segmentation with depth estimation data, arguing that this integration improves obstacle detection compared to using depth data alone.

In contrast, Zhong et al. (2023) [9] proposed a computer vision-based boundary detection system specifically for campus security. They demonstrated the effectiveness of achieving real-time insights and improving campus security predictions through the integration of various security factors. Additionally, Olmos et al. (2018) [10] presented a study aimed at improving weapon detection in surveillance videos using image fusion techniques. Their approach involves applying a series of filters to generate a map that highlights the primary areas of interest. Their work addresses a solution to the real-time pistol detection alarm problem using deep learning CNN-based object detector. They developed a CNN-based classifier on different datasets within the sliding window and region-based detection methods. They developed a new labeled dataset that makes the learning model achieve high detection qualities. The experience in building the new dataset and detector can be useful to guide the development of solutions for other different problems in object detection.

Our proposal focuses on aiding visually impaired individuals with a multi-camera, real-time computer vision system for campus mobility. Unlike prior research on crowd management and weapon detection, it emphasizes precise body point and object detection. It also features robust failure management and obstruction detection, offering a solution for accessibility.

III. Material and Methods

The Multi-Camera Obstruction and Fall Detection System (MCOF)[1] is proposed as a modular pipeline for analyzing real-time video streaming using image processing techniques. This pipeline is divided into several functions, each serving a distinct purpose within the system (see Figure 1). Initially, the model receives the video stream from a set of IP cameras, processes it, and then sends the processed stream to Zoneminder [11] via HTTP requests/responses. The `MCOF` begins by accessing the cameras and performing computer vision tasks using YOLO and other applications for fall and obstruction detection. After processing the video streams, the records are saved to disk. The frame rate (FPS) for recording is adjustable, allowing for customization according to specific needs. Users can set recording criteria, such as time intervals and frame regions, as parameters for the computer vision module. This flexibility ensures that `MCOF` can be adapted to different scenarios and requirements within the university environment.

Processed video streams can be transmitted via HTTP to a designated IP address and port, enabling remote monitoring. The `MCOF` is capable of running this service across multiple nodes, which can receive and store streams as if they were accessing a camera directly. This redundancy provides enhanced security through distributed storage. A key component of the `MCOF` is the `continue_service` module, which initiates and maintains the video streaming service. This module concurrently launches threads for both the video streaming service and the Flask server. This dual-threading approach allows users to view video streams through a web interface while the system continues processing the video in the background.

The main block of the code configures essential variables and parameters for `MCOF` operation, including the IP address, communication port, and background subtraction algorithm (BGS). These parameters are adjustable, allowing the system to be tailored to specific university requirements and various operational contexts. To maintain uninterrupted and reliable

[1]https://github.com/mestrelan/MCOF_IEEE_Healthcom2024

operation, MCOF includes a control structure that automatically restarts the service in the event of a failure. This structure captures exceptions, logs debugging information, and restarts the program if an error occurs. This robust error-handling mechanism ensures that the system remains operational and dependable, even when facing unexpected issues.

The MCOF for university security utilizes advanced computer vision techniques to detect falls and obstructions, enhancing safety for students and staff. Built with various Python modules, each component plays a role in the system's functionality.

A. MCOF Functionality

Input and Output Management: The MCOF begins by using the sys module to handle command-line arguments and manage interaction with the Python runtime environment. This enables efficient read/writing to memory buffers.

Operating System Interaction: The MCOF interacts with the operating system using the os module. This enables essential tasks such as file and directory manipulation, reading environment variables, and managing directories, ensuring dynamic interaction with the filesystem and environment.

Computation and Time Management: The math module enables the MCOF to carry out essential mathematical computations, including geometric transformations and statistical analyses needed for image processing. The time module offers functionalities for measuring execution time, managing date and time values, and creating delays, which are crucial for synchronizing tasks and handling timed events.

Parallel Processing: To manage multiple tasks simultaneously, the MCOF uses the threading module. This facilitates the creation and management of threads, enabling parallel processing. This capability is crucial for allowing multiple web browsers or Zoneminder systems to access the camera stream concurrently.

Command-Line Interface: The argparse module streamlines the creation of a command-line interface, simplifying the configuration and execution of the MCOF with various options and parameters. This enhances user interaction and provides flexibility in configuring the MCOF.

Date and Time Manipulation: The MCOF uses the datetime and time modules to handle date and time information. The datetime module provides classes for manipulating dates and times, while the time module offers functions for working with time values. These tools are essential for timestamping events, scheduling tasks, and managing time-based data.

Error Handling and Debugging: The traceback module captures and prints tracebacks, helping to debug errors and trace the source of exceptions. This feature ensures robust error handling and logging, which are crucial for maintaining reliable system operation.

Data Manipulation and Analysis: The MCOF employs the pandas library for efficient analysis of data structures, tables, and time series data. This is essential for recording and analyzing the MCOF's performance and detection information.

Failure Protection: The MCOF utilizes the subprocess module to manage new processes and link their pipes to exit codes, enabling interaction with external processes and OS command execution. This configuration improves fault tolerance by automatically resetting on errors. BIOS settings ensure the PC restarts after a power loss, and a scheduled time service initiates daily PC boot. A systemd service is set up to launch MCOF during boot. If access issues arise (e.g., network, electrical, or camera failure), the service automatically restarts to restore the camera connection.

HTTP Server for Video Streaming: The MCOF employs the werkzeug.serving and make_server modules to set up an HTTP server using Werkzeug, a WSGI library that manages HTTP requests. This configuration is crucial for streaming processed video over HTTP, making the MCOF accessible to web clients and integrated systems like Zoneminder. The Flask web framework is used to develop the MCOF web application. Flask's lightweight tools, such as Response for sending HTTP responses and render_template for rendering HTML templates, facilitate the display of the video stream on a web page. This setup provides easy access and monitoring through a web browser or video monitoring management system.

B. You Only Look Once – YOLO

The "stream" module manages the real-time processing of video frames. Each frame is captured, pre-processed, and optionally subjected to object detection algorithms such as YOLO and YoloPose [12], [13]. Depending on the model configurations and objectives, the processed frames are either displayed on-screen or recorded in a video file. YoloPose is integrated to identify key body points of individuals. The pretrained weights of YOLOv8 (yolov8n.pt and yolov8n-pose.pt) are utilized for person and object detection. Table I lists the objects identified by the model.

The MCOF processes video frames using the YOLO object detection model to identify objects in each frame. It draws bounding boxes around detected objects, displays the object class and detection confidence, and may also perform a count of detected persons. For each bounding box, the MCOF identifies the coordinates (top-left x1, y1, and bottom-right x2, y2), converts floating-point coordinates to integers for compatibility with OpenCV and extracts the detection confidence.

C. OpenCV

The MCOF utilizes OpenCV [14] to read, record, and manipulate the video stream. It performs various image processing tasks, including filtering, geometric transformations, and edge detection. OpenCV provides a comprehensive set

TABLE I
OBJECTS IDENTIFIED BY THE MODEL.

Category	Objects
People	person
Vehicles	bicycle, car, motorcycle, plane, bus, train, truck, boat
Traffic Signs	traffic light, fire hydrant, stop sign, parking meter
Animals	bird, cat, dog, horse, sheep, cow, elephant, bear, zebra, giraffe
Accessories	backpack, umbrella, bag, tie, suitcase
Sports Equipment	frisbee, skis, snowboard, sports ball, kite, baseball bat, baseball glove, skateboard, surfboard, tennis racket
Kitchen Items	bottle, wine glass, cup, fork, knife, spoon, bowl, banana, apple, sandwich, orange, broccoli, carrot, hot dog, pizza, doughnut, cake
Furniture	chair, sofa, potted plant, bed, dining table, bathroom
Electronics	TV monitor, laptop, mouse, remote control, keyboard, cell phone, microwave, oven, toaster, sink, fridge

of functions for image processing, such as resizing, rotating, cropping, color conversion, filtering, smoothing, and edge highlighting [14]. The `MCOF` defines three main functions for image processing operations: `getKernel`, `getFilter`, and `getBGSubtractor`. The `getKernel` function takes a `KERNEL_TYPE` parameter to specify the type of kernel (structuring element) to return. Kernels are used in morphological operations to define the neighborhood around a pixel. If the type is "dilation," the function creates and returns a 3x3 elliptical kernel using `cv2.getStructuringElement` with `cv2.MORPH_ELLIPSE` [14], [15].

The `getFilter` function performs various morphological operations on an image. It takes an image and a filter type as parameters. If the filter type is "closing," the function applies the closing operation using `cv2.morphologyEx` with the kernel obtained from the `getKernel`("closing") function [14], [15].

The `getBGSubtractor` function creates and returns a background subtractor object based on the type specified by the `BGS_TYPE` parameter. Background subtractors are used to segment moving objects in video [15]. The function returns a KNN background subtractor, created using `cv2.createBackgroundSubtractorKNN` with the specified history parameter [14].

D. MonitoraUFF Project

The `MCOF` is currently being tested in a large video surveillance project, developed and deployed at Fluminense Federal University by the Information and Communication Technology Management Laboratory – GTECCOM[2], a unit of the UFF School of Engineering. The project aims to install monitoring systems across all university buildings and blocks, totaling over 2000 cameras, more than 100 independent monitoring systems, and several control centers integrating multiple systems throughout the campus. For monitoring buildings and their surroundings, the project employs ZoneMinder [11]. Additionally, a proprietary system has been developed for the control center, which will include functionalities such as ticket management. ZoneMinder is an open-source platform for security camera monitoring [11], offering a comprehensive solution for capturing, analyzing, recording, and monitoring CCTV or security cameras on a Linux-based system [11]. By choosing ZoneMinder, the institution ensures a robust, cost-effective, and adaptable solution while maintaining control over its data and infrastructure. The project also aims to remain flexible and utilize free software whenever possible.

[2]https://gteccom.uff.br/

IV. Experimental Evaluation

A. Datasets

We evaluated the model using the NTU RGB+D dataset[3] [16]. This openly available dataset includes 60 action modalities (classes) for activity recognition tasks and comprises 56,880 videos. The tasks are categorized into three groups: Activities of Daily Life (ADL), health-related movements (such as falling), and interactions between people (such as shaking hands). Each entry in the dataset features recordings ranging from 2 to 4 seconds, captured with depth cameras (512×424 resolution), RGB color maps (1920×1080 resolution), and IR cameras (512×424 resolution).

B. Setup

The performance evaluation was conducted on two machines with the following configurations:

Machine 1:

- Debian 12 64-bit operating system
- 1 TB of disk space
- NVIDIA GeForce GTX 1660 graphics card
- 11th Gen Intel Core i7-11700F x 16 processor
- 16 GB RAM
- ASUS PRIME H510M-E motherboard

Machine 2:

- Ubuntu 22.04.4 LTS 64-bit operating system
- 4 TB of disk capacity
- NVIDIA GeForce GTX 1050 Ti graphics card
- Intel Core i7-9700K CPU @ 3.60GHz x 8
- 16 GB RAM
- Gigabyte Z390 Gaming X motherboard

Anaconda was used to manage all tools, with Python v3.8.18 and the following libraries:

- Flask 2.2.2
- ffmpeg 6.1.1
- imutils 0.5.4
- numpy 1.24.4
- pandas 2.0.3
- pytorch 2.2.0
- torchvision 0.17.0

[3]Available at https://rose1.ntu.edu.sg/dataset/actionRecognition/

- werkzeug 2.3.8
- OpenCV 4.8.1

All evaluations leverage the infrastructure provided by the Information and Communication Technologies Management Laboratory (GTECCOM) at Fluminense Federal University. This infrastructure includes a comprehensive wired and wireless computer network, a multi-camera monitoring system, and a centralized control center currently under development. This work supports the extensive university video monitoring project MonitoraUFF, which involves both developing solutions and installing new equipment across UFF campuses, as well as integrating existing equipment. The project promotes accessibility and aligns with mobility expectations from a broad perspective. Additionally, it aims to create a dataset to advance computer vision applications in university security.

C. Fall detection

Figure 4 illustrates a frame after motion detection, background removal, filtering, and body point detection in a video stream. It captures a moment of a fall. The green contour outlines the person's body region, while the red line represents the body axis orientation. Typically, the body axis is vertical when the person is upright and horizontal when lying down. The model detects a fall when there is a sudden shift from vertical to horizontal in the body axis.

We evaluated the `MCOF` on a machine equipped with an NVIDIA GeForce GTX 1050 Ti. The `MCOF` is capable of processing on either the GPU or CPU. Figure 2 shows that without GPU utilization, the application took approximately 1.8 seconds per frame, with GPU utilization at around 0% and GPU memory consumption of 159MB out of 4096MB total. In contrast, Figure 3 illustrates that when Python applications are compiled to leverage the GPU, the processing time per frame is reduced to approximately 0.12 seconds. As shown in Figure 5, GPU utilization increases to about 25%, with GPU memory usage rising to approximately 704MB. Therefore, this GPU can initially handle video streams from up to 4 cameras simultaneously, utilizing approximately 2816MB of the total 4096MB GPU memory.

Next, we evaluated the `MCOF` on a machine with an NVIDIA GeForce GTX 1660. Without GPU utilization, the application requires approximately 7 seconds per frame. Figure 6 shows that when compiled to leverage the GPU, the total processing time per frame is approximately 0.12 seconds, similar to the performance on the previous GPU. Figure 7 indicates that GPU utilization is around 9%, with GPU memory consumption rising to 1528MB out of a total 6144MB.

Initially, it appears feasible to run the application on streams from up to 10 cameras to maximize GPU utilization. However, GPU memory constraints become apparent, as the total required memory (10×1528MB = 15280MB) exceeds the available 6144MB. Consequently, the GPU's memory limitations restrict the simultaneous use of computer vision applications to a maximum of 6 cameras (6×1528MB = 6112MB), which accommodates two more cameras than the previous GPU.

D. Comparison with related work

Nikhil et al. (2021) [5] proposed an object detection system using computer vision techniques, incorporating three distinct modules that apply object detection algorithms and OpenCV to identify rule violations. The system faced challenges due to variations in CCTV camera orientations across different areas, leading to inconsistent results [5]. To address these issues and manage potential obstructions or sudden changes in the camera's field of view (such as if a malicious agent moves the camera), the `MCOF` tracks the date of the last person detected by each camera. If no one passes by within a user-defined timespan, the system generates a ticket indicating that the camera has not detected a person for an extended period, signaling a potential problem or obstruction requiring physical intervention.

Additionally, the system processes a set of frames to calculate a matrix of pixel medians. It then compares the average pixel values of new frames to this matrix. If the average value of a new frame exceeds a predefined threshold, the system generates a ticket for a sudden change in the camera view. To minimize false alarms due to natural variations in lighting, the pixel median matrix is recalculated every two hours.

Ritharson et al. (2022) [6] developed a face recognition system utilizing the HAAR classifier, a color detection algorithm for uniform/dress code detection, and weapon detection in schools using CNN with YOLOv3. In contrast, the `MCOF` aims to enhance university security through computer vision by monitoring human activities. The `MCOF` leverages YOLOv8 [17], integrating it with background subtraction and image filtering techniques. While both studies use computer vision for object identification within a university context, the `MCOF` is designed to classify and recognize a broader range of objects. Guler et al. (2016) [8] describe a real-time intelligent video surveillance system implemented on a GPU. Their experiments, conducted on a PC with an Intel Core i7 CPU and 3.5 GB of usable RAM, utilized an NVIDIA Tesla C2075 GPU with 6.144 GB of total memory. At a resolution of 1,024 × 768, their GPU processed video from 7 cameras in real-time, whereas the CPU could not process any cameras in real-time. The object tracking algorithm achieved a frame processing time of 0.0161 seconds. In comparison, Figure 6 illustrates that the `MCOF`'s object detection starts at approximately 0.023 seconds per frame and reduces to around 0.0060 seconds.

Ashraf et al. (2022) [7] aimed to minimize false negatives and false positives in weapon detection while ensuring fast detection speeds. Their framework also employs YOLO and processes frames with background removal using the Gaussian blur algorithm. The detection speed achieved was 0.010 seconds per frame, significantly faster than Faster R-CNN, which processed frames at 0.17 seconds. The experiments were performed on a 2015 MacBook Pro with 16 GB of RAM, a 256 GB SSD, a 2.8 GHz Intel Core i7 processor, and Intel Iris Pro 1536 MB graphics. Anaconda managed the tools, with Python 3.7.3 and Jupyter Notebook used for implementing the

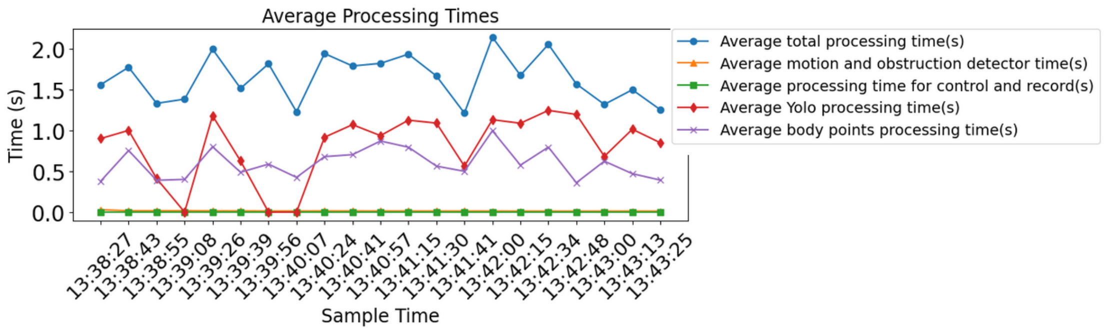

Fig. 2. Average elapsed time without GPU.

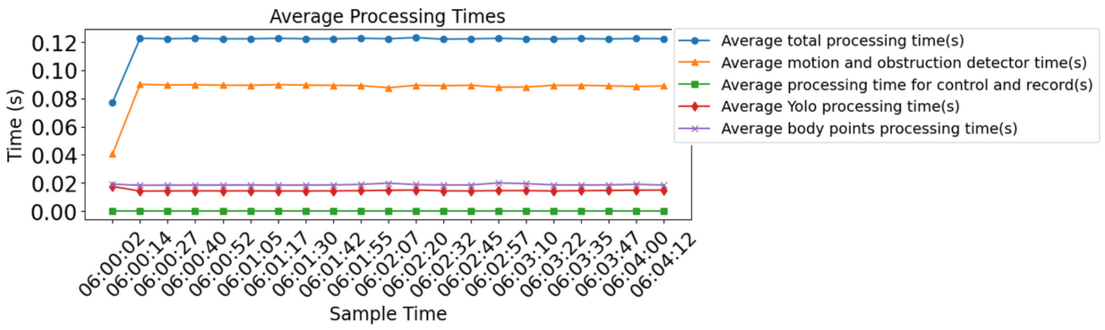

Fig. 3. Average elapsed time using the GTX 1050 Ti GPU and compiled python.

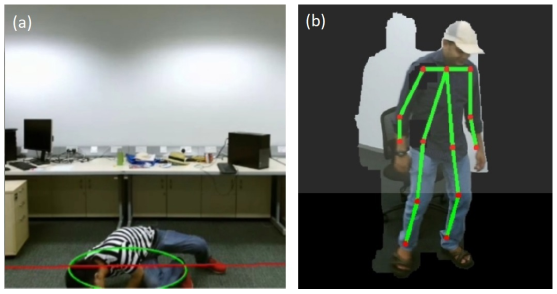

Fig. 4. (a) Frame after fall detection. (b) Frame after motion detection, background removal, filtering and body point detection.

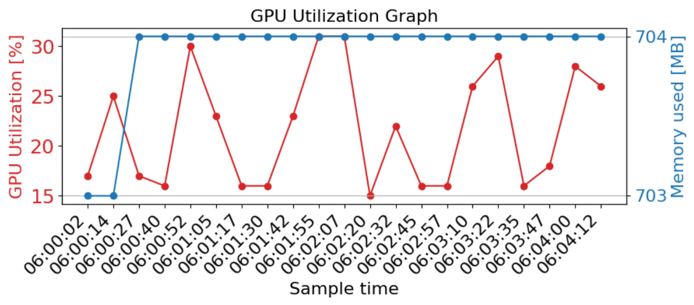

Fig. 5. GPU utilization – GTX 1050 Ti GPU with compiled python.

CNN model. YOLOv5 was implemented on Google Colab with Python 3.6.9 and a 1.85 GB GPU provided by Google. However, reliance on Google's GPU introduces a dependency on Internet connectivity, which could be problematic in real security applications if a connection failure occurs, potentially leaving the environment exposed.

The computer vision model described in [10] achieved a processing time of 0.19 seconds per frame. The Faster R-CNN model [7] recorded a time of 0.17 seconds per frame. In contrast, the computer vision model presented by Ashraf et al. (2022) [7] demonstrated a significantly faster processing time of 0.010 seconds per frame. The model described in this work achieved the following processing times: 0.0118 seconds for detecting body points, 0.0236 seconds for detecting people, objects, and animals, and 0.0514 seconds for detecting movements in real-time camera streaming. Figure 6 illustrates frame time for each examined computer vision model.

V. Conclusions and Future Work

In this paper, we proposed MCOF, a multi-camera, computer vision-based fall and obstruction detection system designed to enhance mobility for visually impaired individuals on university campuses. The MCOF detects falls and sends alerts to the MCOF Control and Record unit, which facilitates further response by the university security team. It also identifies abandoned objects in pathways and corridors, generating alerts for such incidents. The Ticket system categorizes these alerts by time and priority. We assessed the MCOF's ability to manage multiple cameras, given the expansive university environment, and evaluated its processing time to ensure prompt detection of critical alarms for campus security.

Our evaluation compared MCOF processing times with related work in the literature. The results indicated that the system achieves elapsed times similar to those reported by

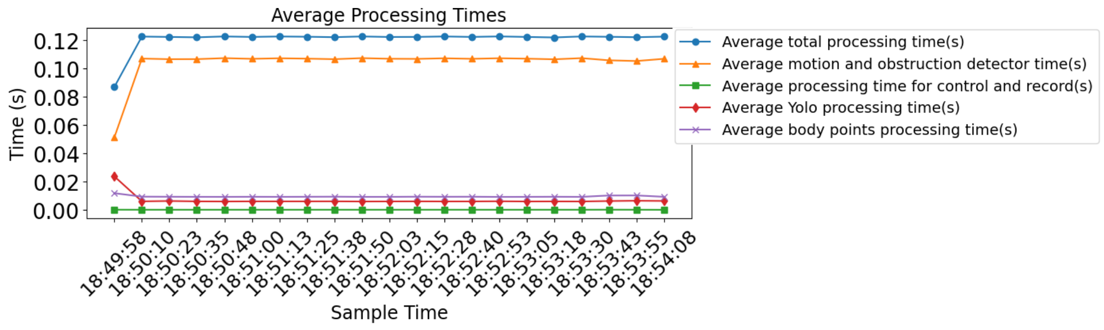

Fig. 6. Frames per second – GTX 1660 GPU with compiled python.

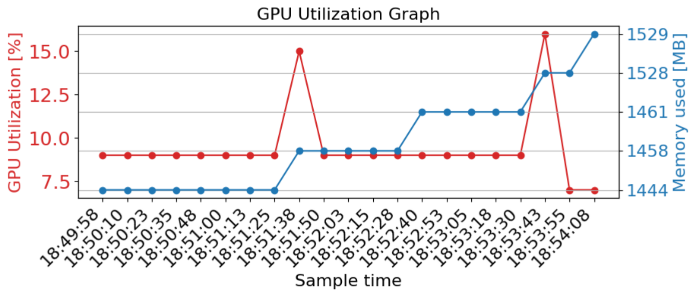

Fig. 7. GPU usage over time – GTX 1660 GPU with compiled python.

Ashraf et al. (2022) [7], but within a multi-camera framework. Specifically, the MCOF recorded 0.0118 seconds for detecting body points. Among the GPUs tested, the GTX 1660 provided the best performance, suggesting its potential to handle simultaneous streams from up to six cameras—two more than the GTX 1050 Ti GPU. This finding is crucial for efficiently managing computational resources in the large-scale MonitoraUFF project, which necessitates deploying multiple GPUs to accommodate the extensive network of cameras.

The MCOF builds upon recent advancements by integrating YOLOv8 with background subtraction and image filtering, aiming to deliver a robust solution for university security. The addition of GPU acceleration and the system's flexibility in processing further amplify its effectiveness, making it a valuable asset for modern security infrastructures. Currently, the MCOF is capable of detecting the fall of a single individual. However, large, fast-moving objects, such as vehicles, can disrupt fall detection. Additionally, crowded environments with numerous background movements can interfere with the system's ability to detect falls accurately. While the MCOF can identify and monitor objects and detect abandoned items based on their duration of presence, it does not yet differentiate between different environmental contexts where these objects are located. Future improvements will focus on enhancing object detection capabilities to further bolster safety and accessibility for individuals with disabilities in university campuses.

References

[1] Froien Farain, Bem Estar - Israeli Home for the Elderly, https://bemestar.froienfarain.org.br/contato/, accessed on June 10 (2022).

[2] U. Vijetha, V. Geetha, Obs-tackle: an obstacle detection system to assist navigation of visually impaired using smartphones, Machine Vision and Applications 35 (2) (2024) 20.

[3] S. R. Flaxman, R. R. Bourne, S. Resnikoff, P. Ackland, T. Braithwaite, M. V. Cicinelli, A. Das, J. B. Jonas, J. Keeffe, J. H. Kempen, et al., Global causes of blindness and distance vision impairment 1990–2020: a systematic review and meta-analysis, The Lancet Global Health 5 (12) (2017) e1221–e1234.

[4] G. Carone, D. Costello, Can europe afford to grow old, Finance and development 43 (3) (2006) 28–31.

[5] N. Raote, M. S. Khan, Z. Siddique, A. K. Tripathy, P. Shaikh, Campus safety and hygiene detection system using computer vision, in: 2021 International Conference on Advances in Computing, Communication, and Control (ICAC3), 2021, pp. 1–7. doi:10.1109/ICAC353642.2021.9697148.

[6] P. Isaac Ritharson, G. Madhavan, M. Rajeswari, D. Brindha, Prevention of school shooting using neural networks and computer vision, in: 2022 Third International Conference on Intelligent Computing Instrumentation and Control Technologies (ICICICT), 2022, pp. 1703–1709. doi:10.1109/ICICICT54557.2022.9917656.

[7] A. H. Ashraf, M. Imran, A. M. Qahtani, A. Alsufyani, O. Almutiry, A. Mahmood, M. Attique, M. Habib, Weapons detection for security and video surveillance using cnn and yolo-v5s, CMC-Comput. Mater. Contin 70 (2022) 2761–2775.

[8] P. Guler, D. Emeksiz, A. Temizel, M. Teke, T. T. Temizel, Real-time multi-camera video analytics system on gpu, Journal of Real-Time Image Processing 11 (2016) 457–472.

[9] B. Zhong, J. B. M. Sharif, S. Salam, C. Ran, Y. Liang, Z. Cheng, Research on optimization of boundary detection and dangerous area warning algorithms based on deep learning in campus security system, Journal of Information Systems Engineering and Management 8 (4) (2023) 22898.

[10] R. Olmos, S. Tabik, F. Herrera, Automatic handgun detection alarm in videos using deep learning, Neurocomputing 275 (2018) 66–72.

[11] R. Stürmer, E. M. Ahlert, Implementação de sistema de videomonitoramento de baixo custo utilizando zoneminder e câmeras ip, Revista Destaques Acadêmicos 12 (4).

[12] D. Maji, S. Nagori, M. Mathew, D. Poddar, Yolo-pose: Enhancing yolo for multi person pose estimation using object keypoint similarity loss, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 2637–2646.

[13] P. Jiang, D. Ergu, F. Liu, Y. Cai, B. Ma, A review of yolo algorithm developments, Procedia computer science 199 (2022) 1066–1073.

[14] G. Bradski, The opencv library., Dr. Dobb's Journal: Software Tools for the Professional Programmer 25 (11) (2000) 120–123.

[15] W. Burger, M. J. Burge, Digital image processing: an algorithmic introduction using Java, Springer, 2016.

[16] A. Shahroudy, J. Liu, T.-T. Ng, G. Wang, Ntu rgb+d: A large scale dataset for 3d human activity analysis, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 1010–1019. doi:10.1109/CVPR.2016.115.

[17] F. N. Ortataş, M. Kaya, Performance evaluation of yolov5, yolov7, and yolov8 models in traffic sign detection, in: 2023 8th International Conference on Computer Science and Engineering (UBMK), 2023, pp. 151–156. doi:10.1109/UBMK59864.2023.10286611.