Intelligent People Flow Monitoring System in Indoor Environments Using YOLO and Cameras

Valdinei Conceição, Aline Souza, Luiz Santos, Paulo Anjos, Ryan Oliveira, Glauco Gonçalves

***Abstract*— Monitoring people flow in indoor environments can help in the improvement of marketing strategies, the optimization of services, and is essential for access control and security in public places. This work addresses the implementation of an intelligent people monitoring system in commercial establishments. Using YOLO technology, the system aims to identify and count individuals in security camera footage. The proposed solution enables real-time detection and data transmission to an MQTT topic, easing the analysis of foot traffic in the monitored areas. Our system evaluations take place in a laboratory located at the Federal University of Pará (UFPA) and demonstrate great results, successfully counting all people in the room.**

***Keywords*— Computer Vision, YOLO, MQTT, Intelligent Monitoring, Traffic analysis, Establishment security.**

I. Introduction

The necessity of monitoring people in indoor environments in an intelligent way is more evident today, since companies and institutions rely heavily on data analysis to determine the best course of actions to take [1]. Understanding behaviors such as the most trodden path, length of stay or rush hours is valuable for creating analyses based on camera data [2].

The increasing demand for intelligent monitoring systems for people flow in indoor environments has been driven by significant advancements in computer vision technologies, generating works like [3]. Recent studies demonstrate the effectiveness of You Only Look Once (YOLO) based models for real-time people detection and counting, offering remarkable accuracy and the ability to operate in different lighting and movement conditions [4]. Such systems are useful not only for security and space management but also for enhancing marketing strategies and resource optimization.

The objective of this paper is to present a people flow monitoring system for indoor environments. The code developed is available at GitHub[1], allowing researchers and developers to contribute, refine, and customize the system to suit their specific needs and use cases.

The system can easily scale with the number of data providers, with the main limitation being local or cloud storage capacity. Similarly, it is possible to add more micro-services to handle tasks like notifying, storing, or processing the data. The system is divided into components, so failures are modular at the architectural level, minimizing the overall impact.

V. Conceição, A. Souza, L. Santos, P. Anjos, R. Oliveira, G. Gonçalves, ITEC, UFPA, Brazil, E-mails: [valdinei.conceicao, aline.sena.souza, luiz.nonato.santos, paulo.anjos, ryan.oliveira]@itec.ufpa.br, glaucogoncalves@ufpa.br. This work was supported by iSACI.

[1]https://github.com/ASTRAson/Nimbus

II. System Details

Figure 1 summarizes the general architecture described in this section. The central component is the Counting Engine, which processes images. The data goes through an MQTT Broker that forwards it to the subscribers, then it is stored in a database and can be visualized on a dashboard.

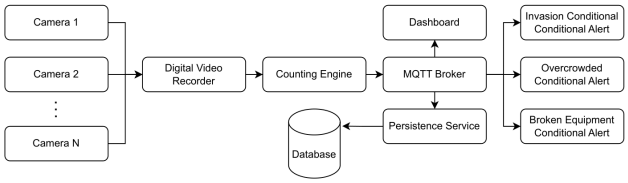

Fig. 1. Example diagram of possible system implementation.

A. Image Capturing and Processing

From an already implemented camera system, we can extract images from the cameras. This task can be made using programming languages as Python with OpenCV library, which is widely used for computer vision applications that provides real-time image and video processing tools.

The images pass through a Counting Engine, to detect everyone that can be seen. This engine can be powered by a YOLO pretrained neural network. Each image is collected every four seconds, and the engine detects each person with a given confidence score, which can be used to filter errors. The number of detected people is temporarily stored, then the mean of the number of countings is sent to an MQTT broker every sixty seconds. This way, possible effects of detection variation caused by movement in the environment are mitigated.

B. MQTT Architecture

1) Broker: It's the central service in the MQTT communication, responsible for exchanging messages between Publishers and Subscribers. In our system it allows a majority of different purposes to the people detection data. One of the most popular brokers available is RabbitMQ [5], known for its robustness and high capacity at managing data.

2) Publishers: We used the image processing system as publishers, where each camera publishes to a specific topic, indicating the room and the camera identifiers. Data is sent every sixty seconds, as specified earlier.

3) Subscribers: The subscribers receive data from publishers, and serve different purposes, such as storing it in the database for later analysis or providing immediate monitoring of the environment's condition.

C. Persistence Micro-service

It is a subscriber in the MQTT architecture, receiving and storing people count data. The service acts as an API, offering endpoints that allow other parts of the system to interact with the data for future access through its CRUD functions.

To implement such micro-service architecture, we can leverage tools like Laravel [6], a powerful PHP framework known for its simplicity, flexibility and efficiency in developing web applications, to manage the persistence layer. Additionally, databases such as MongoDB, Apache Cassandra, Redis or DynamoDB can be utilized to store and manage data efficiently, providing scalability and robustness to the system.

D. Alert Micro-services

Many subscribers can also act as alert micro-services, generating an action when a specific event occurs. Unlike the persistence micro-service, these alerts are triggered immediately for each image sent to the counting engine, allowing for real-time responses. We have tree implemented alerts. Overcrowd: triggered by the excess of people in a location; Intrusion: if the location is closed and shouldn't be anyone there; Broken Equipment: if a publisher is not sending any data to the servers.

III. Evaluation

To validate the system, tests were conducted at LabPRO, a laboratory located at UFPA, with an 22.5m² area that was chosen for its representation of a busy environment. Data was captured using a 1080p webcam positioned at 2 meters in height, providing a clear view of the space.

Image processing was done with YOLOv8 [7], for its simplicity and speed, using the pretained large detection model on a computer with an Intel Core i5-11300H@4.40GHz CPU, GeForce GTX 1650 GPU, and 8GB of RAM. The data was sent to a development MQTT broker, Eclipse Mosquitto [8]. For the persistence micro-service it was utilized the previously mentioned Laravel and MongoDB to store the collected data.

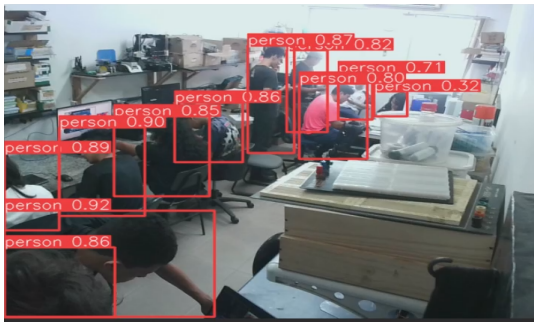

Fig. 2. Validation test at LabPRO's dependencies.

We invited 11 volunteers to walk around the room, Figure 2 shows the system in operation. People are identified by red boxes, with varying confidence scores. The figure shows a minimum confidence score of 0.32, which is enough to detect people, the engine successfully identified all 11 volunteers.

YOLOv8 offers an easy-to-use model, our experiments shows that detection accuracy heavily depends on camera positioning and field of view. Also, overcrowded places or very far away people leads to inaccuracies in counting.

IV. Implementation

The system was implemented with some modifications in March 2024, at the same laboratory used during the validation stage. Running on a Raspberry Pi 4 equipped with a PiCam camera, located at 2.45 meters above the ground, data is collected every thirty seconds and sent to a Thingsboard server, enabling data visualization in a dashboard.

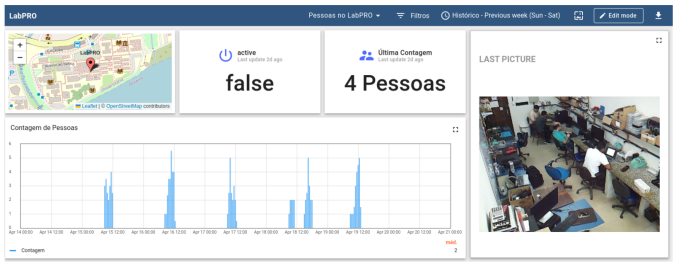

Fig. 3. System implementation at LabPRO's dependencies.

Figure 3 shows a snapshot of the dashboard of the system in operation for approximately one month, along with possible analyses based on the collected data. These data show a daily average of 3 to 4 people in the space simultaneously, but it reached 6 to 7 people at some peak times. Now the system provides greater clarity regarding the usage of the lab, potentially influencing policies of use and operating hours.

V. Conclusion

This work presented a system for monitoring the flow of people in indoor environments, utilizing the YOLO model for real-time detection and counting. This approach enabled the rapid identification of individuals under typical establishment conditions, making the system an effective tool not only for data collection but also for enhancing the overall visitor experience. The data can be used for in-depth analysis and enhancing visitor experiences. Evaluating this data can provide valuable insights that assist in strategic decision-making, such as space planning and implementing security measures.

For future research, it is suggested to add functionalities like facial recognition and behavioral analysis to possibly improve security and offer deeper insights into patterns of movement and activity within the monitored space, potentially increasing the system's utility and applicability in various commercial or institutional settings.

References

[1] Á. Szukits, P. Móricz, "Towards data-driven decision making: the role of analytical culture and centralization eforts" *Review of Managerial Science*, 2023.
[2] O. Järv, R. Ahas, E. Saluveer, B. Derudder, F. Witlox, "Mobile Phones in a Traffic Flow: A Geographical Perspective to Evening Rush Hour Traffic Analysis Using Call Detail Records" *PLOS ONE*, 2012.
[3] W. Wanga, J. Chena, T. Hongb, N. Zhuc, "Occupancy prediction through Markov based feedback recurrent neural network (M-FRNN) algorithm with WiFi probe technology". *Elsevier Building and Environment*. 2018.
[4] D. Reis, J. Hong, J. Kupec and A. Daoudi, "Real-Time Flying Object Detection with YOLOv8", 2023.
[5] RabbitMQ. Available at: https://www.rabbitmq.com/. [Online].
[6] Laravel. Available at: https://laravel.com/. [Online].
[7] G. Jocher, A. Chaurasia, M. Rizwan, *Ultralytics YOLOv8 Docs*. Available at: https://docs.ultralytics.com/. [Online].
[8] Eclipse Mosquitto. Available at: https://mosquitto.org/. [Online].