FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

AI-Driven People Counting on the Edge for Waiting-Time Estimation: A Case Study at FEUP

Vinícius Caparróz Carvalho



Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Prof. Dr. Daniel G. Costa

September 27, 2026

© Vinícius Carvalho, 2026

AI-Driven People Counting on the Edge for Waiting-Time Estimation: A Case Study at FEUP

Vinícius Caparróz Carvalho

Mestrado em Engenharia Eletrotécnica e de Computadores

September 27, 2026

Resumo

A espera em filas é uma experiência quotidiana marcada não apenas pela duração, mas também pela incerteza. Disponibilizar informação contínua sobre o estado de uma fila pode reduzir essa incerteza e apoiar as decisões dos utilizadores. Neste contexto, esta dissertação apresenta um sistema de baixo custo para estimar, em tempo real, a ocupação e o tempo de espera em uma fila, implementado na Faculdade de Engenharia da Universidade do Porto. O sistema foi concebido para operar autonomamente num dispositivo embarcado e preservar a privacidade das pessoas observadas.

Devido à vista parcial e oblíqua proporcionada pelo ponto de montagem autorizado, a instalação foi planeada através de uma reconstrução tridimensional do local, utilizada para selecionar as câmaras, a ótica e a orientação antes da montagem. O sistema utiliza duas câmaras e um modelo de contagem de multidões executado localmente num dispositivo embarcado. A segunda câmara é ativada apenas quando a fila se aproxima do segmento que cobre. As contagens são convertidas em estimativas de tempo de espera com base na teoria de filas e numa taxa de serviço calibrada para a instalação, sendo os resultados publicados num painel de acesso livre. A solução evista gravação de dados pessoais, onde as imagens são processadas apenas em memória volátil durante a inferência, e somente indicadores agregados saem do dispositivo.

Durante a operação, o sistema manteve a cadência de captura ao longo de dias completos de serviço, sem falhas ou intervenção manual, e publicou as estimativas poucos segundos após cada captura. Em comparação com anotações manuais, as contagens apresentaram um erro absoluto médio de 2,93 pessoas e uma tendência sistemática de sobrecontagem. As estimativas de tempo de espera diferiram das medições realizadas no local em menos de um minuto, em média. O campo de visão obtido após a instalação também correspondeu ao previsto na reconstrução tridimensional. Estes resultados demonstram a viabilidade de estimar filas em tempo real com hardware de baixo custo, processamento local e proteção da privacidade, mesmo quando o ponto de instalação oferece apenas uma vista parcial.

Palavras-chave: Estimação do tempo de espera • Contagem de multidões • Computação periférica • Sistemas embebidos • Privacidade desde a conceção • Campus inteligente

Abstract

Waiting in line is shaped not only by duration but also by uncertainty. Providing continuous information about a queue can reduce this uncertainty and support users' decisions. In this context, this dissertation presents a low-cost system for estimating, in real time, the occupancy and waiting time of a queue, implemented at the Faculty of Engineering of the University of Porto. The system was designed to operate autonomously on an embedded device while preserving the privacy of the people being observed.

Because the authorized mounting point provides a partial and oblique view, the installation was planned using a three-dimensional reconstruction of the site, allowing the cameras, optics, and orientations to be selected before deployment. The system uses two cameras and a crowd-counting model running entirely on an embedded device. The second camera is activated only when the queue approaches the segment it covers. Counts are converted into waiting-time estimates using queueing theory and a service rate calibrated for the deployment, and the results are published on an openly accessible dashboard. The solution avoids recording by design, where images are processed only in volatile memory during inference, and only aggregate indicators leave the device.

During operation, the system maintained its capture cadence over full service days without failures or manual intervention and published estimates within seconds of each capture. Compared with manual annotations, the counts achieved a mean absolute error of 2.93 people and showed a systematic overcount. Waiting-time estimates differed from on-site measurements by less than one minute on average. The field of view obtained after installation also matched the one predicted by the three-dimensional reconstruction. These results demonstrate the feasibility of real-time queue estimation using low-cost hardware, on-device processing, and privacy-preserving operation, even when the deployment point provides only a partial view.

Keywords: Waiting-time estimation • Crowd counting • Edge computing • Embedded systems • Privacy by design • Smart campus

Acknowledgements

Primeiramente quero expressar a mais profunda gratidão a minha família. Por seu constante suporte que possibilitou essa jornada, e mesmo a uma longa distância, por sempre terem sua confiança em mim, me auxiliando e dando forças por todo esse percurso.

Quero agradecer meus colegas e amigos, que deram forma, vida e cor a esses anos, que me acompanharam durante esse trajeto, pelo seu companheirismo nos momentos difíceis, e pelos muitos momentos de alegria que proporcionaram. Espero poder retribuir para todos o que proporcionaram para mim, e os desejo o sucesso e força para sempre se reerguerem, e que aproveitem cada momento de suas próprias jornadas. Um agradecimento especial para Gabriel Eggert, Levy Mitsugui, Guilherme de Sarno, Igor Liberato, Estevão Brasil, Enzo Sekito, Enzo Magno, Bernardo Espindola.

Quero especialmente agradecer a minha parceira, Ana Gabriela de Oliveira, cuja confiança em mim me impulsionou a entregar o meu melhor, que me ensina constantemente a apreciar com leveza cada momento, e com sua paciência e carinho transformou até os momentos mais difíceis em memórias a serem apreciadas.

Por fim, quero expressar minha sincera gratidão ao meu orientador, Professor Daniel G. Costa, que me acolheu, me proporcionou a oportunidade de trabalhar ao seu lado, e tornou possível esse trabalho. Agradeço seus conselhos e orientação, que essenciais para esse projeto, e me serviram dentro e fora do campus.

Vinícius Caparróz Carvalho

Declaration of honour

I declare that this dissertation is my own work and has not previously been submitted or assessed at this or any other institution. References to other authors (statements, ideas, thoughts), as well as the use of published works, strictly comply with the rules of attribution and are duly identified in the text and in the bibliographic references, in accordance with the applicable referencing standards. I am aware that the practice of plagiarism or self-plagiarism constitutes an academic offence, as established in the U.Porto Code of Ethics.

I also declare that I have used Artificial Intelligence tools to complete part(s) of this dissertation, and that this use and its results are duly noted and have been carefully checked for errors, inaccuracies, unfounded attributions or other forms of bias in content and arguments, as well as determining authorship.

Vinícius Caparróz Carvalho



United Nations Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework for achieving a better and more sustainable future for all. It includes 17 goals (https://sdgs.un.org/goals) to address the world's most pressing challenges, including poverty, inequality, climate change, environmental degradation, peace, and justice.

This dissertation contributes to this framework through a low-cost, privacy-preserving system that gives a university community real-time information about a shared service, reducing the time lost in its queue and the uncertainty of joining it. The contribution falls under the goals concerning productivity and technological upgrading, accessible infrastructure, and the management of shared urban spaces.

The specific Sustainable Development Goals mentioned have the following names:

SDG 8 Promote sustained, inclusive and sustainable economic growth, full and productive employment and decent work for all

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation

SDG 11 Make cities and human settlements inclusive, safe, resilient and sustainable

SDG	Target	Contribution	Performance Indicators and Metrics
8	8.2	The published waiting time lets a large daily community avoid or reschedule long waits, recovering time that would otherwise be lost in the queue. This productivity gain is obtained through the technological upgrading of an ordinary shared service.	Waiting-time information published continuously during the service window, with estimates deviating from measured waits by under one minute on average.
9	9.1	The system is built from low-cost, widely available hardware, and its planning method eases replication at other sites. Its output is openly accessible to anyone at no cost, extending reliable sensing infrastructure to a scale it does not usually reach.	Unattended operation over full service days without intervention, on single-board hardware, with a publicly reachable dashboard.
11	11.3	The occupancy record supports the data-informed management of a shared campus facility, in the smart campus model, and the zero-recording design shows that this instrumentation can respect the privacy of the people it serves.	Daily records of occupancy and waiting time made available to the institution, with no image stored or transmitted, under a favorable opinion from the University's Data Protection Officer authorizing operation.

"Perfection and power are overrated. I think you are very wise to choose happiness and love"

Iroh, *Avatar: The Last Airbender*

Contents

1 Introduction **1**
- 1.1 Context and Motivation . . . 1
- 1.2 Problem Statement . . . 2
- 1.3 Objectives . . . 3
- 1.4 Contributions . . . 3
- 1.5 Document Structure . . . 4

2 Background and State of the Art **5**
- 2.1 Internet of Things . . . 5
- 2.2 Smart Cities and Smart Campuses . . . 6
- 2.3 Embedded Systems and Edge AI . . . 7
  - 2.3.1 Edge Computing . . . 7
  - 2.3.2 Model Optimization for Constrained Hardware . . . 8
- 2.4 People Counting . . . 9
  - 2.4.1 Non-Visual Sensors . . . 9
  - 2.4.2 Visual Sensors . . . 9
- 2.5 Camera Models and Placement Planning . . . 10
- 2.6 Computer Vision for Crowd Counting . . . 10
- 2.7 Queue Statistics and Waiting-Time Estimation . . . 12
- 2.8 Privacy . . . 14
- 2.9 Literature Review . . . 14

3 Methodology **17**
- 3.1 Case Study . . . 17
- 3.2 Requirements and System Architecture . . . 18
- 3.3 Installation Planning through 3D Reconstruction . . . 21
- 3.4 People Counting Model Selection Protocol . . . 22
- 3.5 Waiting-Time Estimation Model . . . 25
- 3.6 Data Protection Authorization . . . 26
- 3.7 Evaluation Protocol . . . 27

4 Implementation **29**
- 4.1 Development Environment and Tools . . . 29
- 4.2 Hardware . . . 30
- 4.3 Counting Module . . . 32
- 4.4 Estimation and Logging Module . . . 34
- 4.5 Server and Public Dashboard . . . 35
- 4.6 System Integration . . . 36

5 Results and Discussion **38**
5.1 Counting Accuracy 38
5.2 Operation and the Two-Camera Arrangement 39
5.3 On-Device Performance 40
5.4 Waiting-Time Accuracy 41
5.5 Privacy in Operation 42
5.6 Discussion 43

6 Conclusion and Future Work **44**

References **47**

List of Figures

2.1 Schematic representation of the EBC-ZIP crowd-counting pipeline 12

3.1 Schematic layout of the monitored queueing area. 18
3.2 Overview of the system architecture. 20
3.3 3D model of the camera test jig. 21
3.4 Simulated camera views evaluated within the three dimensional reconstruction. . 22
a Reconstructed scene with queue path 22
b Scene populated with human models 22
3.5 Training metrics for the fine tuned YOLO26n head detector over 40 epochs. . . . 25
a mAP@50 25
b F1 score 25
c Precision and recall 25
d Box loss (CIoU) 25
e Classification loss 25
f DFL loss 25
3.6 Fine tuned YOLO26n detections on ShanghaiTech comparing predicted versus ground truth counts across crowd densities [33]. 25

4.1 Prototype case holding the two selected cameras. 31
4.2 3D model of the camera enclosure. 32
4.3 Printed camera enclosure installed at the mounting point. 33
4.4 Example captures from the mounting point. 34
a Full capture from camera 34
b Applied Region of Interest (ROI) to camera view 34
4.5 Public dashboard served by the system. 36

5.1 System count against manually observed ground truth. 39
5.2 Estimated occupancy during the operating period of 26 May 2026. 40
5.3 Published estimate as served to a mobile browser during operation. 42

List of Tables

2.1 Comparison of related people-counting and queue-monitoring systems. 16

3.1 Cross-dataset benchmark of the four pretrained EBC-ZIP models on the Raspberry Pi 5. 24
- a MAE 24
- b RMSE 24
- c NAE 24
- d Images per second 24

4.1 Main software tools, versions, and primary deployment roles. 30

5.1 Counting accuracy in deployment by ground-truth occupancy range. 38
5.2 Operational performance derived from the deployment log. Termination messages are excluded from the estimate and transmission totals. 41
5.3 Waiting-time estimates against measured waits over one session. 42

List of Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
CIoU	Complete Intersection over Union
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSV	Comma-Separated Values
CV	Computer Vision
DFL	Distribution Focal Loss
EBC	Enhanced Blockwise Classification
FEUP	Faculdade de Engenharia da Universidade do Porto
FoV	Field of View
GDPR	General Data Protection Regulation
GPU	Graphics Processing Unit
HAT	Hardware Attached on Top
HQ	High Quality
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
LiDAR	Light Detection and Ranging
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
mAP	mean Average Precision
ML	Machine Learning
NAE	Normalized Absolute Error
PoE	Power over Ethernet
QR	Quick Response
RAM	Random Access Memory
RMSE	Root Mean Square Error
ROI	Region of Interest

SBC Single Board Computer

SDG Sustainable Development Goal

USB Universal Serial Bus

VPN Virtual Private Network

YOLO You Only Look Once

Chapter 1

Introduction

This chapter presents the motivation behind this work, from the acceleration of daily life to the technologies that now allow a physical space to report on itself. It then states the problem that the dissertation addresses, the objective pursued and the goals that support it, and the contributions the work makes. It closes with a description of the structure of the document.

1.1 Context and Motivation

Contemporary daily life is a scenario of constant acceleration, in both social and technological aspects. Social acceleration brings a perception of shortening time, which has become a highly valuable commodity and raised a growing concern with how this resource is spent [1]. This heightened sense of temporal scarcity shifts societal expectations toward immediacy and efficiency, making unmanaged delays a primary source of frustration in daily life. The psychology of waiting describes the causality of such perceptions well, as it shows how the subjective perception of time spent waiting directly impacts individual stress levels and daily well-being, especially when the waiting time is unknown [2].

Technology, in turn, has expanded its frontiers substantially over recent decades, driven by the exponential growth in computing capacity anticipated since the earliest integrated circuits [3]. This progress manifests both in the increased portability and performance of hardware and sensors, and in the recent revolution in Artificial Intelligence (AI), where deep learning enables machines to learn representations directly from raw data and dramatically advances the processing of images, speech, and text [4].

Accompanying these advances, the urban environment is also undergoing a major transformation, embodied by the concept of the smart city, which has increasingly become a standard for urban planning. Smart cities apply technology and data analytics to improve the quality of life of their citizens, raise the efficiency of infrastructure and services, and promote sustainable growth, integrating domains that range from transportation and energy to health and governance [5]. Within this framework, supporting citizens in their daily routines remains a central focus,

driving the search for practical solutions that draw upon the thoughtful instrumentation of physical spaces.

This instrumentation relies on the Internet of Things (IoT) paradigm, which equips everyday environments with sensing, local processing, and communication capabilities. Through this integration, physical spaces cease to be passive and begin to continuously capture and transmit operational data [6]. As low-power embedded processors gained computing capacity and Machine Learning (ML) models became increasingly lightweight and optimized for constrained hardware, running deep learning models locally became technically feasible. This synergy enabled edge AI, an approach where inference is executed directly on local devices rather than on remote servers [7]. Processing data directly at the collection point bypasses the bandwidth limits of streaming raw data to the cloud, while reducing latency and removing the dependency on constant network connectivity [8].

Taking the Faculdade de Engenharia da Universidade do Porto (FEUP) as an environment for technological development aimed at enhancing user experience, aligns with the perspective of a smart campus, an extension of the smart city concept tailored to the smaller ecosystems of university life [9]. Within this setting, this work seeks the conscious instrumentation of the physical environment to promote efficiency, optimize time utilization, and improve overall well-being.

1.2 Problem Statement

Although waiting times in daily routines exert a significant influence on individual well-being and overall quality of life [1, 2], effective monitoring solutions tailored specifically to capture queue dynamics remain underexplored within smart environments.

Existing systems capable of estimating waiting times often rely on central server infrastructure for off device processing, depend on active signal probing from user devices, or require specialized costly hardware. These dependencies make deployment complex and difficult to adapt across diverse physical spaces, demonstrating a lack of simple solutions capable of estimating queue dynamics accurately without requiring extensive supporting infrastructure or active user participation.

Furthermore, real world operational settings, such as a university campus, impose practical limitations that shape implementation from the design phase. Mounting locations cannot be chosen freely, as they depend on the availability of power outlets, structural restrictions, and administrative approvals from local services.

At the same time, the digitalization of the modern world has raised major concerns regarding privacy and data protection, represented in Europe by the General Data Protection Regulation (GDPR), which defends citizens and their rights over personal data. Beyond constraining what deployed monitoring systems may do, regulation requires data protection to be considered by design and by default, shaping how privacy aware systems are conceived [10].

The problem addressed in this dissertation is therefore the absence of an integrated, privacy conscious, and practical monitoring system capable of capturing queue dynamics and estimating waiting times under real world deployment constraints.

1.3 Objectives

By bridging the gap left by existing monitoring systems, this work leverages edge computing to deliver a lightweight, privacy-conscious solution aligned with smart city and smart campus initiatives. To this end, it addresses the main objective: *To design and evaluate an affordable, edge-based monitoring solution capable of estimating queue waiting times and delivering this information to end users in real time.* To achieve this goal, the following specific objectives were defined and pursued:

- Review and map existing people-counting technologies across different sensing paradigms in the literature, evaluating candidate computer vision models directly on the target edge device rather than relying solely on published benchmark metrics.
- Select and configure embedded hardware suitable for physical monitoring, taking into account processing capacity, total cost, and the practical mounting restrictions imposed by real service locations.
- Integrate hardware and software components into a system compliant with GDPR requirements, ensuring that all visual processing occurs locally on the device and that only anonymized, aggregated indicators leave the point of capture.
- Validate the implemented system in a real deployment scenario at the canteen of Building O at FEUP, evaluating queue counting accuracy and waiting-time estimates against ground-truth data collected under actual operational conditions.

The pursuit of these objectives guides this work through the integration of embedded hardware design, computer vision supported by lightweight AI techniques, software engineering, and privacy-by-design principles, offering a practical framework for physical monitoring within smart environments.

1.4 Contributions

In line with the research objectives outlined above, the main contribution of this dissertation is the design, evaluation, and operational deployment of a privacy compliant Edge AI framework for urban monitoring scenario. The developed solution bridges state of the art computer vision models with resource constrained embedded hardware, providing it's processing pipeline suitable for real world applications.

More specifically, this work contributes with:

- A comprehensive review covering sensing modalities, visual counting methods, waiting time estimation, and physical deployment constraints, highlighting existing gaps and development opportunities in smart campus environments.
- An operational AI-driven edge-based system developed using affordable components and deployed at FEUP's Cafeteria, operating under the directives of the GDPR.
- Two public open source repositories containing the software implementation and the deployment configuration, allowing reproducibility.

These contributions support both the theoretical evaluation and the practical execution of edge intelligence in urban environments. The released software allows for community collaboration, future extensions and long term system maintenance.

1.5 Document Structure

This document is set up to take the reader through the development of the work, from its motivation to its results.

The initial Chapter 1 sets the stage by introducing the context in which the study is conducted, the problem it addresses, and the objectives it pursues, giving the reader a sense of why waiting-time information matters and what this dissertation intends to do about it.

The second Chapter 2 serves as the theoretical background surrounding the central theme of the dissertation. It provides the reader with the necessary knowledge on the IoT, embedded systems, and edge AI, along with the techniques available for counting people and estimating waiting times, and it concludes with a review of the related systems found in the literature, positioning the present work within the current state of the art and identifying the gaps it addresses.

The third Chapter 3 presents the design of the system. It walks the reader through the characterization of the deployment site, the requirements and the architecture that follows from them, the planning of the installation, the selection of the counting model, and the formulation adopted for the waiting time, closing with the data-protection appraisal and the protocol prepared for the evaluation.

The fourth Chapter 4 describes how this design was built and put into operation, covering the development environment, the hardware assembled at the mounting point, the software modules that capture, count, estimate, and publish, and the integration that keeps them running unattended.

The fifth Chapter 5 presents the findings of the deployment at FEUP. Through the analysis of the collected data, accompanied by graphical representations of the results, it examines the accuracy of the counting and of the waiting-time estimates, the behavior of the two-camera arrangement, the performance of the device in operation, and the consistency of the data flow with the privacy requirements.

Finally, Chapter 6 concludes the dissertation by summarizing the findings, revisiting the contributions of the work, and suggesting future research directions.

Chapter 2

Background and State of the Art

This chapter provides the fundamental and theoretical knowledge required to understand the context and development of this dissertation. It first sets the scene with the IoT and the smart campus, followed by embedded systems and edge AI, which bring the processing to the edge, together with the techniques that reduce a model enough to run there. It then examines the methods on which this project relies, from the counting of people through different sensing modalities and the placement of the camera that observes them, to the visual counting methods used and the estimation of waiting time from an occupancy count. The privacy requirements that a real deployment imposes are addressed next. A closer focus is finally placed on the related systems that combine these elements in practice.

2.1 Internet of Things

The IoT describes the extension of the Internet to the physical world, in which everyday objects are equipped with sensing, processing, and communication capabilities, becoming able to exchange information about the environment they occupy [6].

Although there is no consensus on a single architecture, systems built under this paradigm are commonly described as having three layers. The perception layer comprises the devices that interact with the environment, acquiring data through sensors and acting upon it through actuators. The network layer transports data using wired or wireless technologies. The application layer interprets the collected data and delivers a service to its consumers, whether a person, an operator, or another system [11, 12].

The technologies supporting this framework have become substantially more accessible, as sensors, processing units, and communication modules have fallen in cost and energy consumption while wireless connectivity has become ubiquitous [11]. As a result, instrumenting a physical space no longer requires prohibitive investment, making applications that were once restricted to industrial settings viable at localized scales, such as individual service points within public or institutional environments.

2.2 Smart Cities and Smart Campuses

The application of connected technologies to urban management falls within the paradigm of smart cities, where data-driven infrastructure improves operational efficiency and elevates the quality of public services. Smart city initiatives integrate information and communication technologies, pervasive sensing architectures, and advanced data analytics directly into the physical fabric of modern urban environments. By embedding computational intelligence into critical municipal services such as public transportation, energy distribution, and civic infrastructure, urban managers can transition from reactive operational models to proactive, data-informed strategies. However, deploying smart city solutions across entire metropolitan areas introduces substantial complexity. Urban environments are characterized by heterogeneous network infrastructure, fragmented administrative oversight, complex legal frameworks governing public surveillance, and diverse citizen demographics [13]. These factors frequently complicate the deployment, systematic testing, and continuous evaluation of emerging urban technologies [14].

At the scale of individual facilities, the same principles are embodied by the smart building, a concept that predates the smart city and has evolved from early building automation systems toward environments that adapt to their occupants. A smart building integrates sensing, processing, and communication technologies to improve energy efficiency, occupant comfort, and operational decision-making, treating the building itself as an instrumented system that can respond to its own state [15]. The benefits attributed to this paradigm include not only reduced energy consumption but also the ability to learn from occupancy patterns and environmental conditions and to adjust equipment accordingly. The literature, however, lacks a universally accepted definition of what constitutes a smart building, and the concept is approached from different perspectives, some centered on energy performance and others on the comfort and safety of the people the building serves [15]. What these perspectives share is the premise that the physical space should be continuously aware of its own usage, which places occupancy quantification at the center of any smart building application.

To address these deployment challenges, the smart campus concept applies these same principles at a smaller spatial scale. University campuses often function as miniature cities, encompassing diverse physical facilities, complex mobility demands, and continuous flows of students, faculty, and staff. Beyond acting as consumers of smart urban services, university campuses serve as living laboratories where emerging technologies can be developed, deployed, and evaluated in real settings. In this context, instrumenting campus spaces provides operational feedback to university administration while directly benefiting the community through improved resource management and daily service transparency [9].

From an architectural perspective, a university campus presents distinct operational characteristics compared to broader urban environments. Campus boundaries, physical layouts, and network infrastructures are centralized under a single institutional administration. This administrative unity streamlines data protection governance, as institutional oversight and designated data protection officers provide clear, predefined frameworks for evaluating and authorizing local sensing

deployments [9]. Consequently, regulatory compliance and privacy safeguards can be integrated directly into the system architecture prior to physical deployment.

Within the broader smart campus literature, physical space instrumentation relies heavily on quantifying occupancy and human flow to align service availability with real-time demand. Rather than relying solely on high-cost infrastructure, recent smart campus deployments increasingly explore low-cost, resource-constrained edge devices to monitor physical spaces while embedding privacy directly into the system architecture. For instance, Neis, Warch, and Hoppe [16] demonstrate how single-board computers and low-cost IoT sensors can simultaneously monitor occupancy and spatial usage, explicitly treating privacy protection as an architectural design constraint rather than a secondary consideration.

Consequently, campus environments have become a useful benchmark for evaluating multimodal sensing and spatial analytics. Monti et al. [17] demonstrated the adaptability of embedded vision systems by monitoring classroom occupancy across diverse spatial layouts using Raspberry Pi hardware. Similarly, Chen et al. [18] evaluated outdoor crowd dynamics using edge-based Light Detection and Ranging (LiDAR), highlighting the capacity of campus living labs to test autonomous, real-time sensing under varying environmental conditions. Ultimately, instrumenting high-frequency shared facilities provides continuous utility because community members repeatedly interact with the same daily services, making real-time occupancy updates directly actionable for individual decision-making.

2.3 Embedded Systems and Edge AI

An embedded system is normally designed around a single set of functions, and combines processor, memory, storage, and the interfaces required to communicate with sensors, actuators, and networks [19]. Software written for such a system is therefore developed against the deployment budgets, and a large part of embedded systems engineering consists of fitting a required function into the resources that the target device makes available, while keeping the system able to run continuously and unattended.

Single Board Computers (SBCs) are a class of embedded platform that already covers most of the components of an embedded system on a single board. They often run a general-purpose operating system, and expose standard interfaces such as Universal Serial Bus (USB), Ethernet, wireless connectivity, and general-purpose input and output pins. These devices are now powerful enough to run non-trivial computation, yet cheap and compact enough to be installed permanently at the point where the data are produced.

2.3.1 Edge Computing

Edge computing is a paradigm in which data are processed near the source that generates them, rather than being transmitted to a remote server for processing [8]. The approach gained importance as IoT deployments grew and the data they generated began to outpace the bandwidth

available to transport them, which made the transmission of raw data to a data centre costly and, for some applications, impractical.

Edge AI, also called edge intelligence, is the particular case of this paradigm in which deep learning models, often after training on stronger hardware, are executed on the edge device, facilitated by an inference framework such as ncnn [7].

2.3.2 Model Optimization for Constrained Hardware

A model trained on desktop or server hardware seldom runs unchanged on an embedded device. The networks that reach the highest accuracy carry tens of millions of parameters and expect a floating-point accelerator, whereas a SBC offers a mobile-class processor, a few gigabytes of memory, and a fixed power budget. Closing this gap is the object of a set of techniques that reduce the size and cost of a model while retaining as much of its accuracy as possible.

The most direct approach is to design the network to be small from the start. Efficient architectures replace the standard convolution with cheaper operations that compute the same kind of feature with fewer parameters. MobileNets [20] factor each convolution into a depthwise step, which filters every input channel on its own, and a pointwise step, which combines the results, reducing the arithmetic of a layer by roughly eight to nine times. Two multipliers then scale the width of the network and the resolution of its input, letting the designer trade accuracy for latency to match the target device. The accuracy of such a network is lower than that of a large model on the same task, but the loss is small relative to the reduction in cost.

Quantization lowers the numerical precision at which a network is stored and evaluated. Weights and activations kept as 32-bit floating-point numbers are represented with fewer bits, typically as 8-bit integers, which reduces the memory the model occupies by a factor of four and replaces floating-point operations with integer arithmetic that a processor without an accelerator runs faster [21]. Converting a model after training is simple but can lose accuracy when the range of values is wide, whereas simulating the reduced precision during training lets the network adapt to it and recovers most of the difference.

Pruning removes the parameters that contribute least to the output. Individual weights near zero can be discarded, which yields a sparse network that occupies less memory but needs sparse-aware libraries to run faster, or whole channels and filters can be removed, which reduces computation directly on ordinary hardware [21]. A large fraction of a network can usually be pruned before its accuracy declines noticeably, and the amount removed becomes a further point on the trade-off between cost and accuracy.

Applying these techniques and running the result on the device is the task of an inference framework. Such a framework converts a model from the format in which it was trained into one the target can execute, and it supplies operators optimized for the processor in place of the general-purpose ones used during training. ncnn is one such framework, built for mobile and embedded processors, free of third-party dependencies, and optimized for the ARM processors that SBCs use [22]. Taken together, an efficient architecture, reduced precision, and a framework that maps the model onto the processor make it feasible to run a modern vision model on portable hardware.

2.4 People Counting

The technologies available for counting people are diverse, and each carries its own trade-offs in cost, accuracy, coverage, and privacy. This section groups them into non-visual and visual sensing, since that distinction largely determines both the privacy profile and the kind of information the sensor can acquire.

2.4.1 Non-Visual Sensors

Non-visual sensors can reduce privacy risks because they generally do not capture directly identifiable imagery [23]. Their limitations instead tend to concern coverage, placement, and the granularity of the information recovered.

Passage sensors, such as active infrared, ultrasound, or similar break-beam devices, measure the flow of people through a controlled opening and are simple and inexpensive, but they only report how many crossed a threshold, and not where those people are in space [24, 23]. Passive infrared sensors, which react to changes in infrared radiation caused by the movement of warm bodies, are equally inexpensive but are better suited to detecting presence than producing an exact count, which requires distributed arrangements [25]. Radar, and mmWave in particular, recovers distance, velocity, and direction from reflections of electromagnetic waves without producing identifiable imagery and independently of illumination [26], but its coverage and resolution remain limited.

At the more sophisticated end of the spectrum, three-dimensional sensors such as LiDAR support long range and high resolution. Chen et al. [18] use LiDAR for outdoor crowd management on a smart campus and report counting accuracy well above 95%, with the further benefit that performance does not degrade in poor lighting. The limitations are mainly that LiDAR remains comparatively expensive, is still susceptible to occlusion in dense scenes, and produces data volumes that demand significant processing capacity.

2.4.2 Visual Sensors

Visual sensors estimate occupancy directly from optical data, capturing rich spatial context and providing sufficient detail to monitor defined regions of interest. Unlike passage sensors, which only track directional crossings at a single point, camera-based sensing can quantify how many individuals are simultaneously present within an observed area.

These devices encompass several modal variations. Standard RGB cameras are widely available, cost effective, and capable of monitoring broad areas, though their reliance on ambient light makes them sensitive to shadows and illumination changes. Depth-sensing RGB-D cameras incorporate active infrared emission or stereo optics to measure distance, offering explicit spatial boundaries that simplify target separation in dense environments. Thermal cameras capture surface temperature distributions, enabling light-independent operation and inherently protecting personal identity, though at higher hardware costs and reduced spatial resolution.

Mounting geometry also shapes sensor performance. Top-down overhead arrangements minimize mutual occlusion between individuals and simplify counting logic, but restrict the monitored area per sensor. Conversely, oblique or perspective view angles offer broader spatial coverage from a single point, but introduce scale variations and severe occlusion as individuals overlap along the camera Field of View (FoV). Because visual sensors directly produce information in the form of pixel arrays, converting these optical signals into derived data requires computational processing, creating a direct dependency on Computer Vision (CV) techniques [27].

2.5 Camera Models and Placement Planning

The relationship between a camera and the scene it captures is described by the pinhole camera model, which maps points in the three-dimensional world onto the two-dimensional image plane through the camera's intrinsic parameters, namely its focal length, principal point, and sensor dimensions [28]. Among these, the focal length together with the sensor size defines the angular extent of the observable scene, known as the FoV. A shorter focal length, or a wider lens, yields a larger FoV that captures more of the scene at the cost of angular resolution, whereas a longer focal length narrows the FoV and concentrates resolution on a smaller region. For a fixed mounting position, this trade-off directly governs how much of a queue a camera can observe and at what level of detail.

Viewpoint, mounting height, and incidence angle affect occlusion between people, the apparent size of distant subjects, and the extent to which the region of interest is covered, all of which have been identified as decisive factors for the practical accuracy of vision-based counting [29]. Determining a suitable configuration is therefore a design problem in its own right, addressed in the literature under camera placement and view planning, where the objective is to select positions and orientations that maximize coverage of the target region while minimizing occlusion and unfavorable viewing angles.

Establishing such a configuration through physical trial and error is costly, particularly when the installation point is constrained and access is limited. A more efficient alternative is to reason about placement within a virtual reconstruction of the environment, where candidate positions and camera parameters can be evaluated before any hardware is mounted. Parameswaran, Shet, and Ramesh [29] demonstrated the value of this approach, using a virtual environment to study how deployment conditions such as viewpoint and illumination affect counting performance.

2.6 Computer Vision for Crowd Counting

The use of visual sensors for people counting is a well-established domain within CV. Early approaches relied on handcrafted image-processing techniques, such as background subtraction, foreground segmentation, and kernel-based filtering, to estimate regions occupied by people [30]. With the emergence of Convolutional Neural Networks (CNNs), crowd counting transitioned from

manual visual cues to learned feature representations, substantially improving robustness against clutter, perspective distortion, and high crowd density [31, 32].

Modern CNN-based crowd counting methods are typically trained using point-annotated head locations. During training, these annotations serve as supervision signals to guide the generation of spatial density or count representations [33]. The model backbone extracts feature maps from the input image, encoding visual patterns related to heads, crowd textures, and background elements. A specialized prediction head converts these feature maps into spatial count maps, which can be visualized as heatmaps where higher values indicate denser gatherings. Summing the predicted values across the entire spatial map yields the total count estimate [34].

Rather than regressing continuous density values directly, Liu et al. [35] proposed framing counting as a blockwise classification task. In this formulation, the input image is divided into localized blocks, and the model predicts a probability distribution over discrete count classes for each block. This converts count regression into a classification problem while retaining numerical output by computing the expected value across predicted distributions. This paradigm was subsequently refined by the Enhanced Blockwise Classification (EBC) framework, which uses integer-valued count bins and an auxiliary density-map loss to improve count precision [36].

The EBC-ZIP architecture extends this blockwise formulation to better model the statistical sparsity of local crowd data. Because crowd images contain large unpopulated areas, most blocks contain zero people while populated blocks contain small discrete counts. EBC-ZIP addresses this through a Zero-Inflated Poisson formulation, where one branch estimates the probability that a block is empty while another predicts the count rate for occupied regions. Combining these estimates suppresses false positives in background areas while maintaining accuracy in populated regions [37]. Figure 2.1 illustrates this pipeline from input image to final count prediction.

The blockwise formulation is also well suited to constrained hardware. Combined with a lightweight backbone, an EBC-ZIP counting head provides a practical route to on-device inference on embedded platforms [37].

Alternatively, You Only Look Once (YOLO) provides a detection-based approach suited for embedded hardware, localizing individuals through discrete bounding boxes rather than continuous density distributions. As a single-stage detector, it frames object detection as a unified regression task from input pixels to spatial bounding boxes and class probabilities in a single forward pass, enabling real-time operation [38].

This makes YOLO an effective baseline for on-device counting. Detection-based methods perform reliably in sparse to moderately crowded environments, though accuracy degrades in dense scenes due to target overlap and severe occlusion. Detecting heads rather than full bodies mitigates occlusion in crowded scenes [39], though dedicated density estimation models remain better suited for high-density crowds. Comparative studies confirm this trade-off, showing strong performance at lower crowd densities and reduced accuracy as scene density increases [40, 41].

[1]The input-image example was adapted from the ShanghaiTech dataset [33].

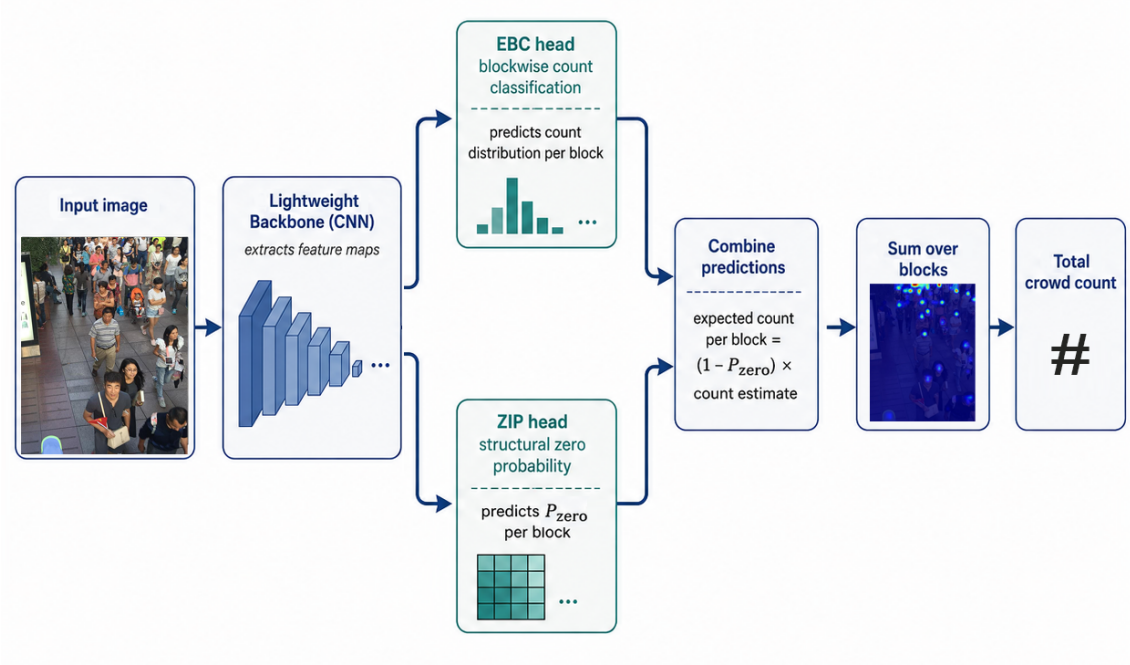


Figure 2.1: Schematic representation of the EBC-ZIP crowd-counting pipeline[1]. Figure generated with the assistance of ChatGPT and manually reviewed by the author.

2.7 Queue Statistics and Waiting-Time Estimation

Estimating how long a person will wait in a queue can be approached in several ways. In infrastructure-based systems, the problem is commonly divided into two stages: estimating the number of people present and translating that occupancy into a waiting-time estimate.

Within each stage the proposed techniques diverge considerably. Parameswaran, Shet, and Ramesh [29] rely on classical CV techniques to detect and count people from surveillance imagery, which works best on top-down views, and report average absolute errors of 1.72 and 3.25 people from their high-ceiling viewing points. They also laid groundwork for reasoning about the practical variability of real deployments, such as queue geometry, counter type, illumination, and camera viewpoint, through simulation in a virtual environment. Rajvanshi and Pramod [42], working in a supermarket checkout setting, update the approach by combining CV with a deep neural network, estimating not only the number of customers in line but also the items in each basket and the cashier's throughput in order to predict waiting time more precisely.

A second family of approaches dispenses with fixed infrastructure altogether and infers queue dynamics from signals emitted by the users' own smartphones. Wang et al. [43, 44] monitor Wi-Fi signals from a single point near the service area to recover the moments at which a person reaches the head of the queue and finishes being served, resolving service and waiting times to within roughly ten seconds. Wang et al. [45] propose a mobile crowdsensing system that detects queuing behavior from accelerometer data and ambient context in line-based queues, while using manual reporting in ticket-based queue scenarios. These methods are attractive because they require little dedicated hardware, but they assume that the people in the queue carry, and are willing to share

data from, a suitable device.

These systems do not derive their estimate from an explicit model of the queue; the wait is produced directly from a count, a throughput figure, or a pair of recovered timestamps. Queueing theory provides the analytical foundation for relating physical occupancy to waiting time in service systems. At the core of classical queueing analysis is Little's Law, which establishes a fundamental relationship between the long-run average number of customers in a system (L), the average arrival rate (λ), and the average time spent in the system (W):

$$L = \lambda W. \tag{2.1}$$

While Little's Law characterizes steady-state system averages over extended horizons [46], real-time sensing yields instantaneous point-in-time occupancy measurements. Translating a single spatial count into an expected waiting duration requires mapping discrete occupancy to temporal dynamics. Queueing theory supplies this link by formalizing assumptions regarding service rate, arrival distributions, and queue discipline [47].

A widely used analytical framework is the $M/M/c$ queue family, which models Markovian arrival processes, exponentially distributed service times, and c parallel servers [48]. For a single effective service channel ($c = 1$), the $M/M/1$ model is parameterized by the arrival rate λ and the service rate μ. System utilization is defined as:

$$\rho = \frac{\lambda}{\mu}. \tag{2.2}$$

A steady-state solution exists when $\rho < 1$, indicating that service capacity exceeds the average arrival rate ($\lambda < \mu$). Under stationary conditions, the expected total number of customers in the system (L) and the expected number of customers in the queue (L_q) are expressed as:

$$L = \frac{\lambda}{\mu - \lambda}, \quad L_q = \frac{\lambda^2}{\mu(\mu - \lambda)}. \tag{2.3}$$

Similarly, the corresponding expected total time in the system (W) and the expected waiting time in the queue (W_q) are given by:

$$W = \frac{1}{\mu - \lambda}, \quad W_q = \frac{\lambda}{\mu(\mu - \lambda)}. \tag{2.4}$$

Distinguishing between L and L_q (as well as W and W_q) is critical when parameterizing queueing models. While L includes both individuals waiting in line and the person currently being served, L_q accounts strictly for those waiting in the queue. The selection between these formulations depends on whether physical sensing covers the entire service facility or isolates the waiting area.

Although the classical $M/M/1$ model provides a tractable formulation, real service points frequently experience non-stationary arrival patterns, time-varying service rates, and finite buffer capacities. Advanced queueing formulations, such as non-homogeneous Poisson processes or

data-driven service rate estimation, address these limitations by modeling dynamic fluctuations while preserving analytical interpretability [48].

2.8 Privacy

A system that processes images of people in a public space falls within the scope of data-protection regulation. In the European Union, the GDPR governs the handling of personal data, and images from which an individual can be identified are explicitly treated as personal data under Article 4(1) [49].

Rather than merely constraining what systems may do, the GDPR has actively shaped how privacy-aware systems are designed [10]. Article 5 establishes the core principles that any processing must satisfy, among them lawfulness, fairness and transparency, data minimization, and storage limitation. The principle of data minimization requires that the data collected be limited to what is strictly necessary for the intended purpose, while storage limitation restricts how long it may be retained. Article 25 reinforces this with the requirement of *data protection by design and by default*, under which privacy safeguards must be built into the architecture from the outset rather than added afterwards.

These constraints have motivated approaches such as edge-based video analytics, in which computation is performed on or near the capturing device and raw imagery need not be transmitted to an external server [50]. Processing images locally and transmitting only aggregate indicators can reduce the exposure of personal data and support the principles of data minimization and storage limitation. Local processing alone does not establish regulatory compliance, but it provides an architectural safeguard that can be combined with access control, limited retention, transparency, and formal institutional assessment.

2.9 Literature Review

This section examines how the elements previously described have been combined in practice, reviewing systems that count people or estimate occupancy in shared spaces, with attention to the sensing modality they adopt, the hardware on which they run, the way they handle privacy, and the extent to which they were validated in a real deployment.

The works discussed here were collected from IEEE Xplore, the MDPI journals, ScienceDirect, and arXiv, using combinations of the terms *people counting*, *crowd counting*, *occupancy estimation*, *queue*, *smart campus*, *edge*, and *privacy*. Priority was given to systems evaluated in real environments and to those deployed on constrained hardware, since these are the conditions under which the present work operates. Table 2.1 summarizes the reviewed systems.

A first group of works observe queues directly, predicting its length or the time a person will wait. Parameswaran, Shet, and Ramesh [29] detect and count people from surveillance imagery using classical computer vision, reporting average absolute errors of 1.72 and 3.25 people from

high-ceiling viewpoints. Rajvanshi and Pramod [42] extend the estimate in a supermarket checkout setting, combining computer vision with a deep neural network to account for the complexity of shopping baskets and the throughput of the cashier, so as to predict the waiting time more precisely. Wang et al. [43, 44] recover service and waiting times to within roughly ten seconds from Wi-Fi signals observed at a single point, while Wang et al. [45] infer queuing behaviour from smartphone accelerometer data and ambient context. These works address the estimation problem in depth, but the camera-based ones process imagery off the device, and the smartphone-based ones assume that the people in the queue carry and share data from a suitable device.

With special attention to privacy and processing on constrained hardware, Monti et al. [17] count the people in university classrooms using cameras attached to Raspberry Pi boards, applying transfer learning so the system can move between rooms of different characteristics without a dedicated training phase, and report a maximum mean absolute error of 1.23 people. Chen et al. [18] address outdoor crowd management on a smart campus with LiDAR, reporting counting accuracy above 95% and the further benefit that performance does not degrade in poor lighting, at the cost of a more expensive sensor and larger data volumes.

Finally, Determe et al. [51] monitor crowds in a public building through the passive detection of Wi-Fi probe requests, deriving a mathematical model of the count and validating it against a camera-based system over nine weeks of data. Cokbas, Ishwar, and Konrad [52] count people with an overhead low-resolution thermal sensor at a doorway, tracking entries and exits through the crossing of a virtual line. Ahamed et al. [53] count people from an overhead camera on a low-power edge computer, combining a fine-tuned detector with a dedicated tracking and counting algorithm to reach 97% accuracy at 20 to 27 frames per second. Xie et al. [54] compares several efficient deep learning architectures for counting people on an ultra-low-resolution infrared array, deploying them on a microcontroller and characterizing each not only by accuracy but by the memory, latency, and energy it requires on the target device.

The present work is positioned relative to these systems. As in Monti et al. [17] and Ahamed et al. [53], the counting model runs on a single board computer; beyond counting, it converts the counts into a waiting-time estimate through a queueing model. Previous works that estimate waiting times either process imagery off the device or depend on the devices carried by the user, whereas here no user participation is needed and images are handled only in volatile memory, so that no imagery is stored or leaves the device, for privacy and practicality. The hardware is built around a SBC and two camera modules, without the expensive sensors such as LiDAR. The system was validated in operation, where the counts showed a mean absolute error of 2.93 people and the waiting-time estimates differed from the measured waits by less than one minute on average.

Table 2.1: Comparison of related people-counting and queue-monitoring systems.

Work	**Scenario**	**Sensor / Platform**	**Advantages**	**Disadvantages**
Monti et al. [17]	Smart campus classroom	Camera + Raspberry Pi	On-device processing; transfers between rooms without retraining; MAE $\leq$ 1.23 person	Counts only; room occupancy, not queue
Determe et al. [51]	Public building / library	Wi-Fi probe requests	No imagery captured; accuracy modeled analytically; extensive validation	Counts only; broad-area count; depends on users' devices
Parameswaran, Shet, and Ramesh [29]	Retail queue	Camera (high viewpoint)	Simulation-based placement planning; queue-focused	Off-device processing; requires top-down view
Rajvanshi and Pramod [42]	Supermarket checkout	Camera	Rich waiting-time model	Off-device processing; infrastructure-dependent
Wang et al. [43, 44]	Service queue	Wi-Fi (single point)	Minimal hardware; times to within $\sim$10 s	Requires users' devices; no spatial detail
Wang et al. [45]	Line / ticket queues	Smartphone crowdsensing	No fixed infrastructure	Requires user participation
Chen et al. [18]	Smart campus (outdoor)	LiDAR	Accuracy $>$ 95%; light-invariant	Counts only; costly; large data volume; occlusion in dense scenes
Cokbas, Ishwar, and Konrad [52]	Smart building doorway	Low-resolution thermal	On-device processing; privacy by low resolution; public dataset	Counts only; crossings, not occupancy of a region
Ahamed et al. [53]	Smart building / transport	Camera + edge computer	On-device processing; 97% accuracy at 20–27 FPS	Counts only; no waiting-time estimate
Xie et al. [54]	Indoor occupancy	Infrared array + microcontroller	On-device processing; deploy-aware model selection; very low memory and energy	Counts only; 8×8 resolution, cannot resolve queue geometry
Present work	Smart campus canteen	Two cameras + Raspberry Pi 5	On-device processing; privacy aware design; waiting-time estimate published in real time; MAE 2.93 people	Partial view of the queue; fixed time estimation formula

Chapter 3

Methodology

This chapter describes the design of the system developed to estimate queue occupancy and waiting time. It begins with the characterization of the deployment site, which establishes the constraints that the architecture and the installation had to satisfy. The requirements and the resulting architecture are presented next, followed by the study that determined where the cameras could be placed and how much of the queue each of them would cover. The protocol used to select the counting model and the formulation adopted to convert an occupancy count into a waiting time are then described. The chapter closes with the data-protection appraisal that authorized the deployment and with the protocol prepared to evaluate it.

3.1 Case Study

Deploying an IoT solution for instrumenting the environment requires a comprehensive characterization of its operational constraints. The deployment site for this study is the main dining facility situated within Building O at the Faculty of Engineering of the University of Porto. This canteen serves students, faculty members, and administrative staff during working days within a fixed service window running from 11.30 to 14.00.

The physical layout of the queue forms an L shaped geometry dictated by the structure around it. The primary queue corridor extends along a straight section, directing pedestrians toward the food service counters. Under high demand, the queue turns at a corner and extends into a secondary lateral hallway, as demonstrated in Figure 3.1. The corner is shaped by walls, which creates a visual barrier, obstructing a continuous line of sight of the full queue trajectory from most viewing points.

Another challenge that appears in this environment is the presence of an adjacent parallel queue operating within the same physical space. This second line consists of a queue formed on a secondary service offered, running parallel to the main corridor. Because both groups share the same open hall, the automated vision system must explicitly exclude people in the secondary line from the primary queue count to prevent false positive detections and inaccurate occupancy estimations.

Based on architectural floor plans available at the institutional website and spatial measurements taken on site, point A in Figure 3.1 was initially identified as an optimal mounting location for monitoring the entire queue, offering an elevated position with an unobstructed line of sight covering both the main corridor and the lateral extension of the queue. However, consulting with FEUPs technical service revealed that point A lacked local power infrastructure and network connectivity, rendering continuous powered operation impossible without structural modifications. Consequently, sensor deployment was restricted to the authorized point B, equipped with accessible power outlets and mounting support. Although point B provides clear coverage of the primary corridor, it suffers from limited visual capacity during peak demand periods when the queue extends beyond the corner into the unobserved lateral hallway.

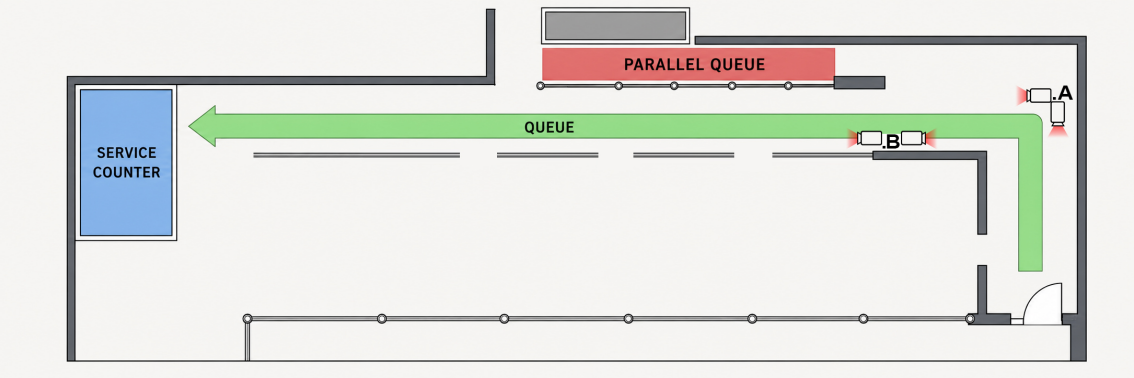


Figure 3.1: Schematic layout of the monitored queueing area within Building O. Points A and B represent the optimal and feasible mounting points, respectively. Figure generated with the assistance of ChatGPT and manually reviewed by the author.

Further analysis of the queue dynamics revealed a distinct operational contrast between the arrival rate λ and the service rate μ, from the described formulas utilized for waiting time estimation as established in Section 2.7. Observations conducted across different sections at the operating window demonstrated that the arrival rate fluctuates significantly throughout the service period. In contrast, the service rate μ remains relatively stable, yielding an observed mean service rate μ of 2.85 people per minute, with minimal variance between sessions. Accordingly, this mean value is incorporated into the queue estimator as a calibrated constant, while the computer vision pipeline dynamically tracks the fluctuating queue occupancy.

3.2 Requirements and System Architecture

To develop an edge computing device capable of fulfilling the objectives established in Section 1.3, a set of operational and functional requirements were set to guide the overall system design. First, the physical deployment should work unattended, with daily operation during service hours, operating reliably without manual supervision or daily local intervention. Second, to protect user privacy and strictly comply with data protection regulations, the architecture must enforce zero

exposure of identifiable data of the system's users. Third, the system must maintain low operational latency by processing and publishing updated queue estimates, ensuring that published waiting times accurately reflect current queue conditions. Finally, physical installation must remain straightforward, guided by available power and network infrastructure.

To achieve these design goals, a modular software structure was adopted following the system decomposition principles proposed by Parnas [55]. Under this approach, system responsibilities are partitioned into isolated modules that encapsulate specific tasks and hide implementation details behind clear interfaces. The processes running locally on the edge device were divided into two independent software modules, accompanied by a remote server, as illustrated in Figure 3.2, which hosts the public dashboard and maintains historical records. These structural components correspond directly to the three layers of the general internet of things architecture described in Section 2.1. Sensors acquire visual data at the perception layer, the local institutional network transports processed occupancy payloads across the network layer, and the remote web server presents public metrics at the application layer. Furthermore, the complete source code is publicly available across two dedicated repositories covering the edge software and the dashboard web application [56, 57]. This modular decomposition allows individual software components to be updated, tested, and maintained independently while preventing localized runtime faults from propagating across the overall system.

At the edge device level, both software modules execute as independent system services managed by `systemd`, the standard service manager of the Linux operating system. Configuration files set the system to automatically handle process lifecycle management by launching the edge services at opening hours, continuously supervising execution state, and automatically restarting software modules in the event of unexpected runtime terminations [58]. Delegating lifecycle management to `systemd` provides autonomous supervision required for unattended physical deployment, enabling recovery from transient hardware or software faults without requiring regular manual operator intervention. In addition, relying on standard service management makes all process startup, execution, shutdown, and error recovery events fully explicit and auditable through centralized operating system logs.

The first edge module handles video frame capture and neural network inference, initializing automatically when the corresponding service becomes active. When a new frame is acquired from the camera, it undergoes Region of Interest (ROI) filtering and remains exclusively in volatile Random Access Memory (RAM) during object detection and count estimation, as detailed in Section 4.3. The raw image buffer is immediately overwritten and deleted from memory after inference completes, guaranteeing that no visual data or identifiable imagery is ever written to non volatile memory. The resulting queue occupancy count is then written atomically to a structured JavaScript Object Notation (JSON) status file, `status.json`, which serves as the primary data exchange interface between the two local modules. This file based communication decoupling is a simple form of inter process communication and allows the inference engine to run at its own execution rate without blocking or depending on downstream tasks.

The second edge module is started prior to the inference service in order to monitor the status

file continuously for state changes. Upon detecting a newly written occupancy value in the status file, this module executes the waiting time estimation algorithm described in Section **??**. Once the estimation calculation is complete, the module formats the result and transmits the updated state payload to the central web server over the local network infrastructure. Restricting this second module to lightweight mathematical calculations and network communication ensures that data transmission reacts immediately to fresh estimates without competing for computational resources needed by the computer vision module.

At the conclusion of daily operating hours, `systemd` initiates an automated shutdown sequence by terminating the inference module, which writes a final closing status payload to the shared JSON file prior to exiting. Upon detecting this explicit termination signal, the second module executes its own shutdown routine, transmitting summary metrics and daily operational logs to the central server before terminating its process. This structured shutdown protocol prevents data accumulation on local edge storage and consolidates all operational logs on the remote server. A detailed description of the remote web server, database storage, and public visualization dashboard is presented in Section 4.5.

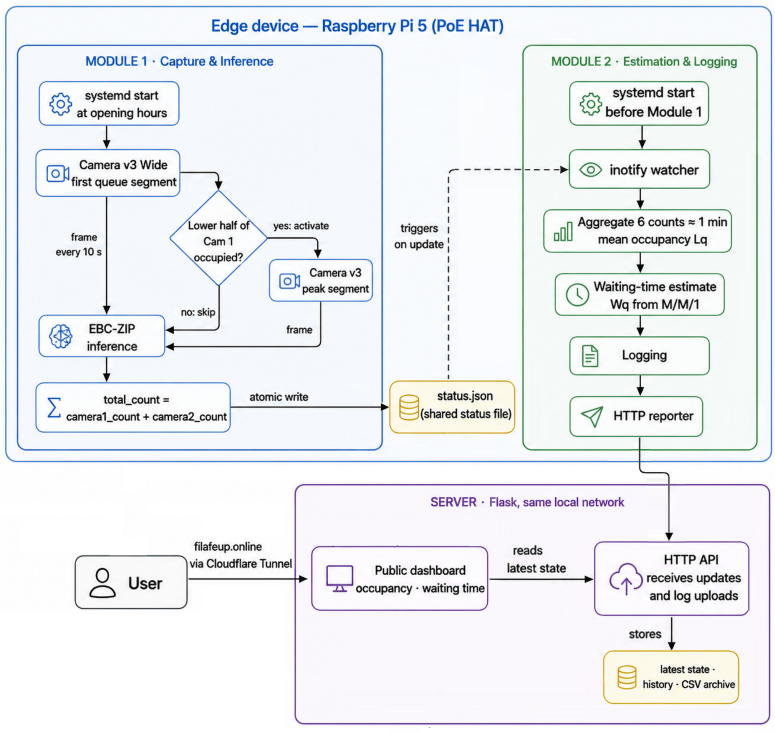


Figure 3.2: Overview of the system architecture. Figure generated with the assistance of ChatGPT and manually reviewed by the author.

3.3 Installation Planning through 3D Reconstruction

The physical constraints at the monitored site dictate the spatial coverage attainable by the visual capture system. Because sensor placement was restricted to the single authorized mounting location, evaluating candidate camera hardware and optical configurations required study to ensure adequate coverage before physical deployment. Rather than conducting empirical trial and error mountings on site, which would incur operational disruption and unnecessary hardware alteration, a comprehensive spatial planning methodology was established using virtual three dimensional reconstruction.

The virtual environment was constructed in Blender, an open source three dimensional modeling software. Spatial geometry for the model was achieved through parameters from architectural floor plans provided by the university facility services and physical measurements recorded on site. Each candidate camera hardware was modeled within the software environment by configuring synthetic camera objects according to the manufacturer's datasheet specifications, such as sensor of the Raspberry Pi Camera Module v3, Camera Module v3 Wide, and High Quality Camera with an 16mm lens. The virtual cameras were placed at the exact coordinates of the authorized mounting bracket to evaluate spatial perspective projection and angular coverage across the entire queue path.

To validate these sensors and establish a functional baseline, a custom physical support was built (Figure 3.3). It was then showed that the cameras behaved similarly in performance under the same image resolution, leaving the sensor selection to a FoV comparison.

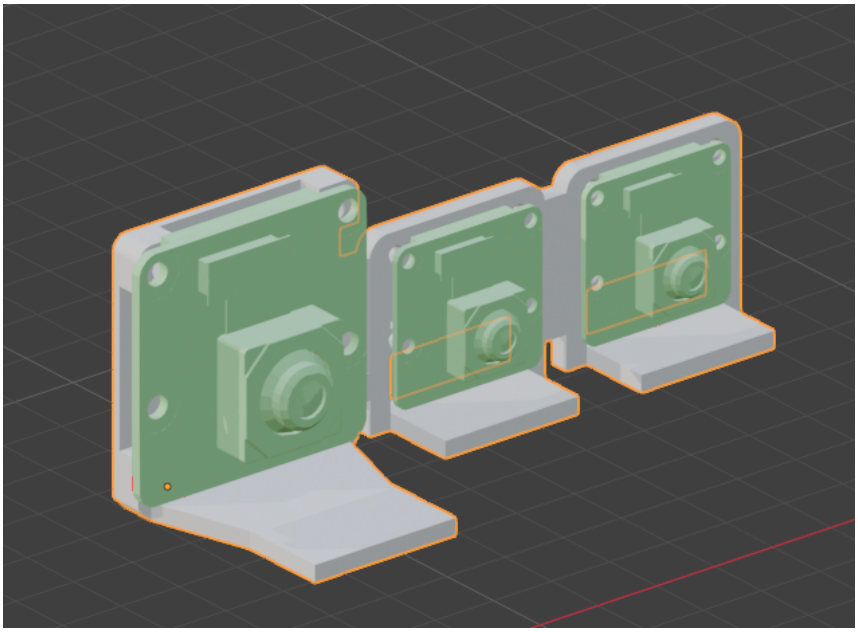
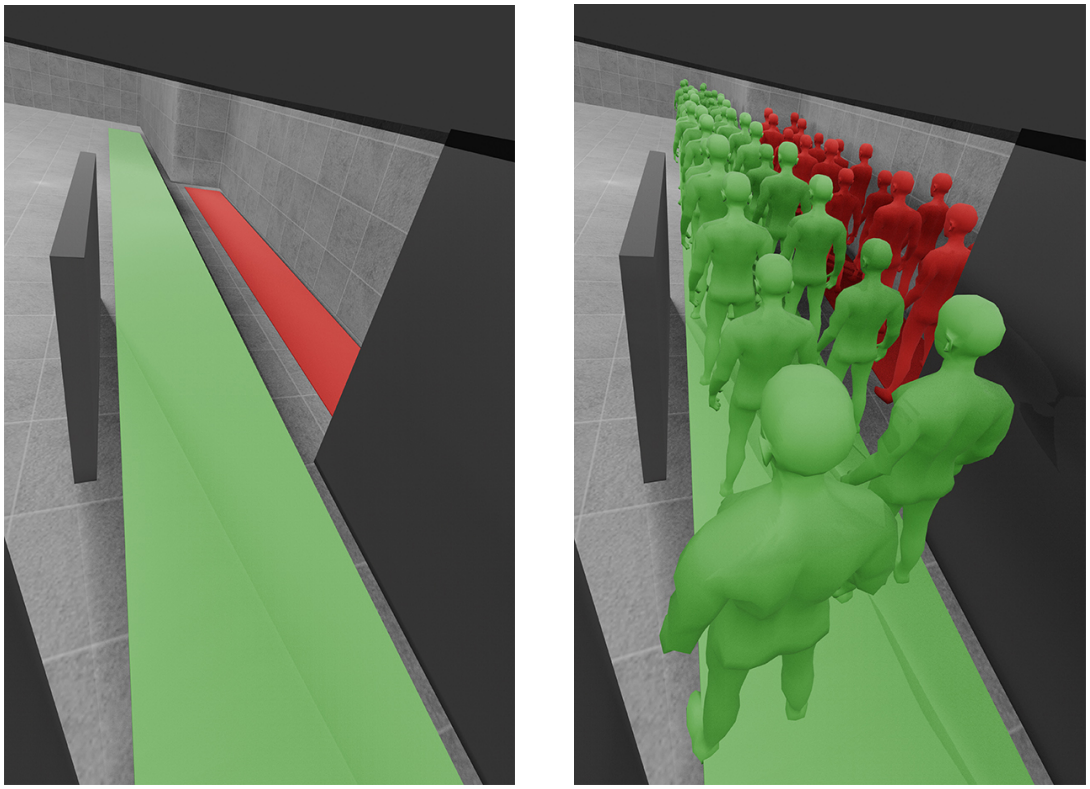


Figure 3.3: 3D model of the camera test jig.

The 3D environment analysis revealed that combining the wide angle camera with the default v3 model provided full visibility of the primary queue segment while minimizing visual blind

spots, as demonstrated on Figure 3.4. Furthermore, the virtual model enabled the determination of required mounting angles to achieve the intended coverage, which were subsequently integrated into the CAD design of the custom 3D printed enclosure as described in Section **??**, ensuring that the physical enclosure permanently fixed the optics at the intended orientation upon physical installation minimizing post deployment adjustments.



(a) Reconstructed scene with queue path (b) Scene populated with human models[1]

Figure 3.4: Simulated camera views evaluated within the three dimensional reconstruction.

3.4 People Counting Model Selection Protocol

Building upon the literature review presented in Chapter 2, two distinct visual counting methodologies were selected as candidates for the edge processing module. These candidates represent the two primary paradigms in automated crowd analysis, namely blockwise density regression represented by EBC ZIP and single stage object detection represented by YOLO.

Deploying neural models on resource constrained hardware requires dedicated framework optimization. Although the lightweight variant of EBC ZIP contains 0.81 M parameters, its reference implementation relies on PyTorch, an execution framework designed primarily for server class graphics hardware and dynamic graph computation. Direct deployment of PyTorch models on mobile processors introduces significant memory overhead and unoptimized Central Processing

[1]Human model adapted from the Free3D (https://free3d.com/3d-model/rigged-male-human-442626.html).

Unit (CPU) instruction execution. To achieve efficient processing on the Raspberry Pi 5 hardware, all candidate models were converted to ncnn, an open source inference framework optimized specifically for mobile class CPU architectures, described in Section 2.3.2.

The official distribution of EBC ZIP includes four pretrained model variants corresponding to four distinct crowd counting benchmark datasets [37]. Although the ShanghaiTech B dataset matches the physical density profile of the target canteen environment, these pretrained variants differ substantially in their underlying training configurations. The models incorporate distinct crop resolutions, internal spatial scaling factors, and preprocessing requirements, which directly influence runtime latency and memory consumption. As a consequence, selecting the optimal variant required empirical hardware benchmarking directly on the Raspberry Pi 5 platform rather than relying solely on theoretical dataset domain alignment.

Following successful conversion to ncnn through the official framework's repository , all four pretrained EBC ZIP models were systematically evaluated across the official validation splits of four crowd counting benchmarks, comprising ShanghaiTech A (182 images), ShanghaiTech B (316 images), QNRF (334 images), and NWPU (500 images). To maintain evaluation integrity, the experimental benchmarking conditions strictly replicated the reference standards defined in the original EBC ZIP repository, with inference routines executed utilizing four ncnn computing threads configured in light mode and model input dimensions followed the official repository setup, applying 448×448 pixels for the ShanghaiTech variants, 672×672 pixels for QNRF, and 896×896 pixels for NWPU.

Table 3.1 outlines the cross dataset benchmark results used to establish the candidate selection criteria, and show that the converted models achieved equivalent results to their original versions when applied to their own datasets. The quantitative evaluation demonstrates clear trade offs between spatial resolution, numerical precision, and operational speed. While the model trained on NWPU achieved high counting accuracy across several dense splits, its elevated input resolution of 896×896 pixels resulted in substantial computational latency, rendering real time deployment infeasible on the target hardware. ShanghaiTech model variants provided higher processing rates but suffered from severe accuracy degradation when evaluated on other datasets. The QNRF trained model variant provided the most effective balance between density generalization and computational efficiency, maintaining consistent error metrics across diverse crowd structures while operating at significantly higher frame rates than the NWPU model. Consequently, the QNRF variant was selected as the representative EBC ZIP model for embedded deployment.

To establish a comparative evaluation against detection based approaches, a lightweight object detector was evaluated under the same operational assumptions. As reviewed in Chapter 2, real time object detection frameworks such as YOLO offer suitable execution speeds for embedded targets, although standard person detection models suffer from rapid precision degradation when applied to dense crowds where inter person occlusions occur. To mitigate bounding box overlap and target occlusion, a compact YOLO26n network was fine tuned specifically for single class head detection, following the methodological setup described by Vadavadagi et al. [39]. Because the published literature did not completely specify all hyperparameter values, unstated

Table 3.1: Cross-dataset benchmark of the four pretrained EBC-ZIP models on the Raspberry Pi 5. Rows indicate the dataset each model was trained on, columns the dataset it was evaluated on.

(a) MAE

	Dataset			
Model	**SHA**	**QNRF**	**SHB**	**NWPU**
SHA	71.00	141.53	19.40	173.88
QNRF	79.93	96.28	13.64	115.56
SHB	177.32	290.91	8.25	182.58
NWPU	89.44	104.61	9.03	66.85

(b) RMSE

	Dataset			
Model	**SHA**	**QNRF**	**SHB**	**NWPU**
SHA	114.00	263.48	35.02	530.60
QNRF	131.92	161.86	23.30	411.24
SHB	305.85	555.32	12.79	782.05
NWPU	146.19	199.41	15.24	413.32

(c) NAE

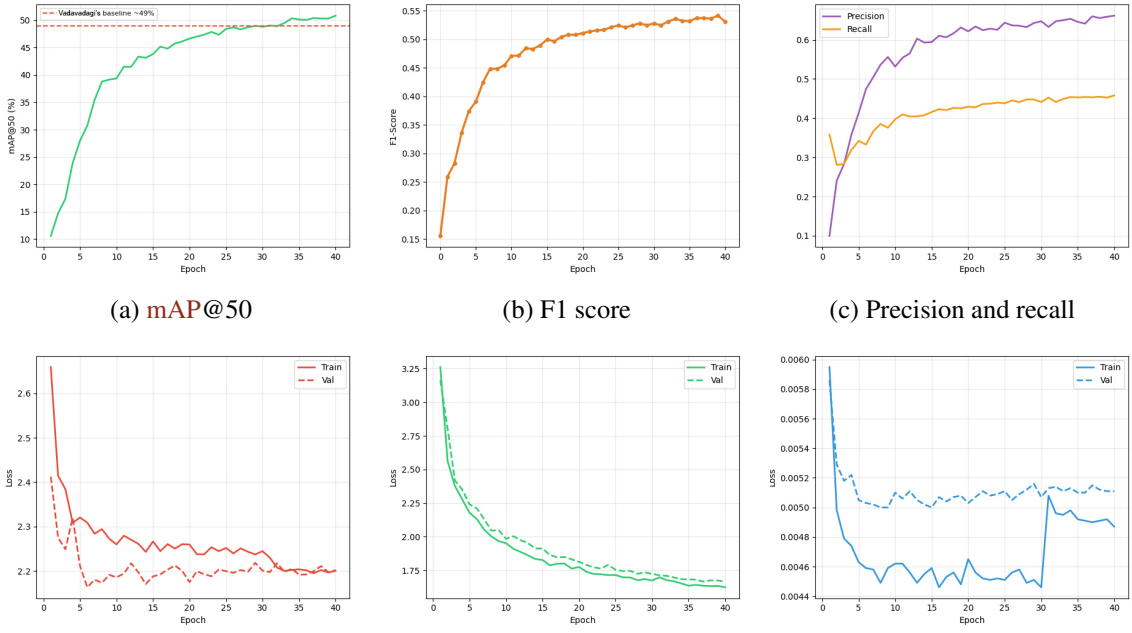
	Dataset			
Model	**SHA**	**QNRF**	**SHB**	**NWPU**
SHA	16.8%	21.0%	17.0%	275.6%
QNRF	18.3%	15.2%	10.9%	247.9%
SHB	35.6%	33.6%	7.2%	58.7%
NWPU	20.8%	15.6%	7.3%	18.1%

(d) Images per second

	Dataset			
Model	**SHA**	**QNRF**	**SHB**	**NWPU**
SHA	9.55	1.92	4.20	1.10
QNRF	5.46	3.51	5.39	1.41
SHB	4.57	1.94	9.45	1.12
NWPU	3.03	1.89	3.42	1.32

training configurations were aligned with standard Ultralytics settings [59] and established detection practices [60]. The model was trained over 40 epochs. Figure 3.5 depicts the progression of optimization metrics throughout training, including bounding box loss using CIoU, classification loss, distribution focal loss, overall precision, recall, and mean Average Precision (mAP)@50.

When evaluated on the ShanghaiTech B validation dataset, the fine tuned head detector achieved an Mean Absolute Error (MAE) of 77.06, an Root Mean Square Error (RMSE) of 113.00, and an Normalized Absolute Error (NAE) of 0.51, operating at an average throughput of 2.19 images per second. Figure 3.6 illustrates visual results obtained across varying crowd densities, highlighting the model's high variability on it's results. Comparing these empirical outcomes directly against the performance of the QNRF trained EBC ZIP model presented in Table 3.1 confirms the clear superiority of blockwise density regression for embedded deployment. While the fine tuned YOLO26n detector provided localized bounding box predictions, its counting accuracy degraded significantly under moderate crowd density, and its processing speed remained lower than that of EBC ZIP QNRF. Consequently, the QNRF variant of EBC ZIP was selected as the counting engine for final edge integration.



(a) mAP@50 (b) F1 score (c) Precision and recall

(d) Box loss (CIoU) (e) Classification loss (f) DFL loss

Figure 3.5: Training metrics for the fine tuned YOLO26n head detector over 40 epochs.

3.5 Waiting-Time Estimation Model

The estimation module converts each visual count into an expected waiting time. To formulate a consistent queueing estimator, the system relies on $M/M/1$ queueing relations. The camera system observes the queue within a defined Region of Interest. Because this region strictly encompasses people standing in line, the detected count corresponds to queue occupancy L_q rather than total system occupancy L, which would otherwise include the individual currently being served at the counter. The service rate μ is treated as a calibrated constant established during the case studies in Section 3.1, whereas the arrival rate λ cannot be measured directly from an instantaneous frame snapshot but can be recovered analytically by inverting the standard $M/M/1$ equilibrium equation for queue length.

In an $M/M/1$ system, the relationship between queue occupancy L_q, arrival rate λ, and service

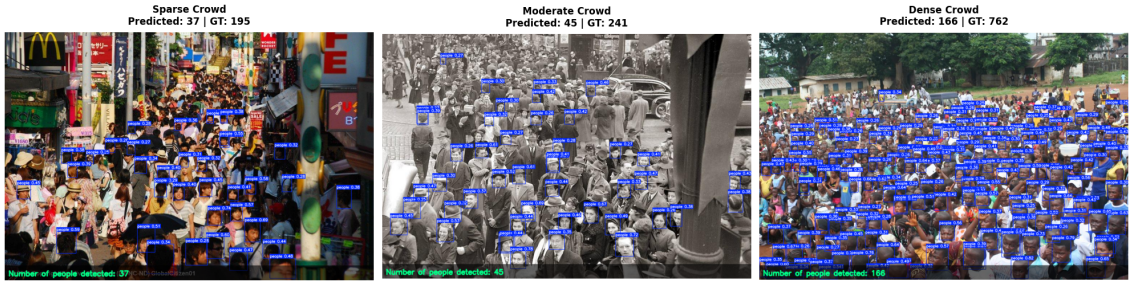


Figure 3.6: Fine tuned YOLO26n detections on ShanghaiTech comparing predicted versus ground truth counts across crowd densities [33].

rate μ follows Equation 2.3. Solving for λ as a function of L_q yields

$$\lambda = \frac{\mu\left(\sqrt{L_q^2 + 4L_q} - L_q\right)}{2}. \tag{3.1}$$

Substituting this recovered arrival rate into the expression for expected queue waiting time $W_q = \frac{L_q}{\lambda}$ yields

$$W_q = \frac{L_q + \sqrt{L_q^2 + 4L_q}}{2\mu}. \tag{3.2}$$

This result expresses expected waiting time strictly in terms of the two parameters available to the system, namely the occupancy L_q captured by the vision pipeline and the calibrated service rate μ.

The $M/M/1$ formulation provides a closed form mathematical relationship, but an underlying theoretical assumption is not fully satisfied in a physical dining facility, it being that Poisson arrivals assume a stationary arrival rate λ over time, whereas physical canteen traffic experiences non stationary bursts during peak hours.

To compensate for instantaneous image noise, temporary body occlusions, and transient arrival spikes, the estimation module does not output raw frame values directly. Instead, the system maintains a rolling window across six consecutive measurement frames and computes their arithmetic mean. This six frame filtering window smooths out momentary visual counting errors and high frequency fluctuations while preserving overall responsiveness to true queue dynamics.

3.6 Data Protection Authorization

Since the system processes images of people in a university space, its deployment was subject to formal appraisal by the University of Porto's Data Protection Unit, with FEUP acting as the data controller under the GDPR. The appraisal examined the eventual constitution of datasets from captured images, the technical safeguards ensuring their immediate discard, the duty of information towards data subjects, the institutional authorization for the installation, the characteristics and positioning of the cameras, the temporal operation of the system, and who has access to the data produced.

The process shaped the final design. The constitution of a local image dataset, considered in the initial proposal, was abandoned, images are used exclusively and transitorily for immediate inference, held in volatile memory and discarded as soon as the count is extracted, with only aggregate indicators retained. The duty of information is fulfilled through signage at the site, bearing a camera symbol and a Quick Response (QR) code linking to the dashboard, where the full description of the processing is available, with the final texts provided by the Data Protection Officer.

The physical installation was authorized by the faculty's technical services, and the processing itself by the direction of FEUP. A favorable opinion was issued, conditioned on the system

operating exclusively as described, and on any substantial change to its operation, or continuation beyond the project's term, being subject to renewed appraisal.

3.7 Evaluation Protocol

Counting and waiting time accuracy were evaluated during live operation at the Building O canteen on 26 May 2026. Ground truth occupancy was collected through direct observation from a position adjacent to the camera mounting location, providing approximately the same perspective and spatial coverage as the monitored ROI. Manual counts were recorded in real time and matched by timestamp with the corresponding system outputs. This process produced 100 paired observations, with ground truth occupancies ranging from 2 to 17 people.

In parallel, real waiting times were empirically measured during the same operational session. The timing protocol involved tracking individual queue members from the instant they joined the queue until they reached the counter service point to receive their meal. A total of 15 individual patron waiting durations were recorded. These measured waiting times were compared directly with the estimates published by the system at corresponding one minute intervals.

Counting accuracy was evaluated using five quantitative metrics, namely MAE, RMSE, NAE, mean bias, and the Pearson correlation coefficient. Let y_i denote the ground truth observation for sample i, $\hat{y}_i$ represent the corresponding system output, and N be the total number of paired observations.

The MAE measures average discrepancy magnitude and is expressed as

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|. \tag{3.3}$$

The RMSE penalizes larger errors more heavily, reflecting sensitivity to sporadic estimation spikes, defined as

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}. \tag{3.4}$$

The NAE normalizes each error by the corresponding ground-truth count, giving a relative measure that is comparable across scenes of different densities,

$$\text{NAE} = \frac{1}{N}\sum_{i=1}^{N}\frac{|\hat{y}_i - y_i|}{y_i}. \tag{3.5}$$

Systemic deviation toward overestimation or underestimation is captured by the mean bias, formulated as

$$\text{Bias} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i). \tag{3.6}$$

Under the sign convention adopted in this study, a positive bias signifies system overestimation, whereas a negative bias denotes system underestimation relative to physical reality.

Linear tracking performance across paired samples is measured by the Pearson correlation coefficient, defined as

$$r = \frac{\sum_{i=1}^{N}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2 \sum_{i=1}^{N}(\hat{y}_i - \bar{\hat{y}})^2}}, \tag{3.7}$$

where $\bar{y}$ and $\bar{\hat{y}}$ represent the arithmetic sample means of ground truth observations and estimated outputs, respectively.

Waiting time performance was assessed using MAE, RMSE, and mean bias expressed in minutes. Correlation metrics were omitted for waiting times because the number of directly measured waiting durations gathered in a single session remains too limited for correlation or determination coefficients to yield statistical significance. A full statistical characterization across a larger collection of physical measurements is deferred to future work.

System operational reliability was evaluated using historical Comma-Separated Values (CSV) log files covering 24 operational sessions between 25 May and 25 June 2026. These logs contain 3,251 one minute records and were analyzed to evaluate output cadence, data transmission success, processing latency, secondary camera activation events, processor thermal dynamics, and memory footprint.

Chapter 4

Implementation

This chapter describes how the methodology presented in the previous chapter was considered when building the expected system. It begins by discussing the environments in which the work was carried out and the tools used in each of them, followed by the hardware assembling details. The software is then described in a data-driven fashion, from the capture and counting of frames on the device, through the estimation and logging of waiting times, to the server that publishes them. The chapter closes with the integration of these modules into a system that runs unattended throughout the service period.

4.1 Development Environment and Tools

The implementation of the system relies on an operational workflow distributed across different computational environments to balance processing requirements and deployment tasks. Model training for the computer vision tasks was executed on a Google Colab runtime [61] with an NVIDIA Tesla T4 Graphics Processing Unit (GPU) and 15.6 GB of dedicated memory reported by the runtime. The virtual scene rendering and spatial placement planning described in Section 3.3 were performed in Blender [62] on a local workstation. The main application server ran on a machine with an Intel Xeon Silver 4214R CPU at 2.40 GHz and 32 GB of RAM. Code development was conducted mainly on the target device itself, which runs a 64-bit operating system and handles image acquisition, model inference, local management, and remote data transmission.

The software pipeline integrates specialized libraries across the computational setup to ensure efficient execution. On the edge device, picamera2 handles hardware accelerated image acquisition directly from the camera sensor. Vision models developed and trained with Ultralytics [59] were converted and executed using the ncnn inference framework to optimize memory utilization and processing speed on ARM architecture. Operational interfaces were served locally through a lightweight Flask web application, while cloudflared provided secure communication to external services without requiring open inbound ports or static network configurations.

Table 4.1 presents the software tools, version specifications, and primary roles within the workflow.

Table 4.1: Main software tools, versions, and primary deployment roles.

Tool	**Version**	**Primary Role**
Raspberry Pi OS	Bookworm (64-bit)	Linux operating system for edge node host execution and management.
Python	3.11	Module integration, pipeline orchestration, and system service execution.
picamera2	0.3.2	Frame acquisition, sensor configuration, and camera control.
ncnn	20230816	Neural network inference execution on ARM CPU architecture.
Ultralytics	8.4.0	Object detection model training, fine-tuning, and weight export.
Blender	4.5 LTS	Physical site 3D reconstruction, camera placement planning, and FoV simulation.
Flask	3.0.0	Backend API endpoint hosting and web dashboard serving.
cloudflared	2024.2.0	Encrypted HTTPS tunneling for local endpoints without port forwarding.

The selection of these software tools and computational platforms was guided by the need to balance processing throughput with operational efficiency. Offloading heavy training workloads to cloud infrastructure and dedicated server hardware ensured fast model development without overburdening edge resources. At the same time, deploying lightweight web frameworks and dedicated inference engines on the edge device enabled real time processing and secure remote management while maintaining system stability.

4.2 Hardware

The device must run continuous CNN inference using the target architectures detailed in Chapter 2. This requirement excludes typical microcontroller platforms, whose memory and processing capacity fall orders of magnitude below what these multi-camera vision tasks demand. At the opposite end, boards with dedicated AI accelerators and small form factor computers offer more processing capacity, but at a higher cost and power draw than the application justifies. Between the two, SBCs provide the processor and memory capacity sufficient for the lightweight inference techniques reviewed in Section 2.3.1, at a cost compatible with a single service point.

Within this class, the Raspberry Pi 5 presents itself as a viable choice for an initial prototype, even if it sits above the basic needs of the developed system. This overshoot is justified by the development it enables, as its broad support and documentation lower the complexity of building the system, its multiple camera ports accommodate a wide range of options, and the availability of compatible Power over Ethernet (PoE) hardware resolves the power delivery that Section 3.1 shows to be constrained at the mounting point. Selecting a board with margin at this stage also removes hardware capacity as a variable, so that the effort stays on the system rather than on fitting it to a constrained device.

The central single board computer is a Raspberry Pi 5 equipped with 8 GB of Random Access Memory (RAM). At the time of project implementation, this unit carried a cost of approximately 80 euros, though subsequent supply chain constraints and elevated hardware demand have since impacted market availability and baseline pricing. Visual acquisition is supported by three candidate camera modules evaluated during the preliminary design stage, including a Raspberry Pi Camera Module v3 priced at roughly 25 euros, a Raspberry Pi Camera Module v3 Wide priced at 35 euros, and a Raspberry Pi High Quality (HQ) Camera costing 50 euros paired with a 16 mm C mount lens priced at 50 euros. Electrical power and network communications are unified using a Power over Ethernet (PoE) Hardware Attached on Top (HAT) costing approximately 25 euros, which incorporates an integrated heat sink to ensure stable operation during sustained computational workload. The combined hardware expenditure for the complete multi camera evaluation setup remains under 230 euros, while the final deployed dual camera node requires an investment below 170 euros.

The three cameras differ primarily in their optics and their resulting FoV. The Camera Module v3 and its Wide variant share the same underlying sensor technology but cover distinct angular extents, while the HQ Camera paired with the 16 mm lens provides a narrower view with higher spatial resolution. Together they span a range of FoV from which the one best matched to the monitored area can be chosen, through the study described in Section 3.2.

Before the final enclosure, a simpler prototype case was built to hold the two selected cameras in a fixed relative position, shown in Figure 4.1, allowing for the development of the dual camera capture and counting software, before installation.

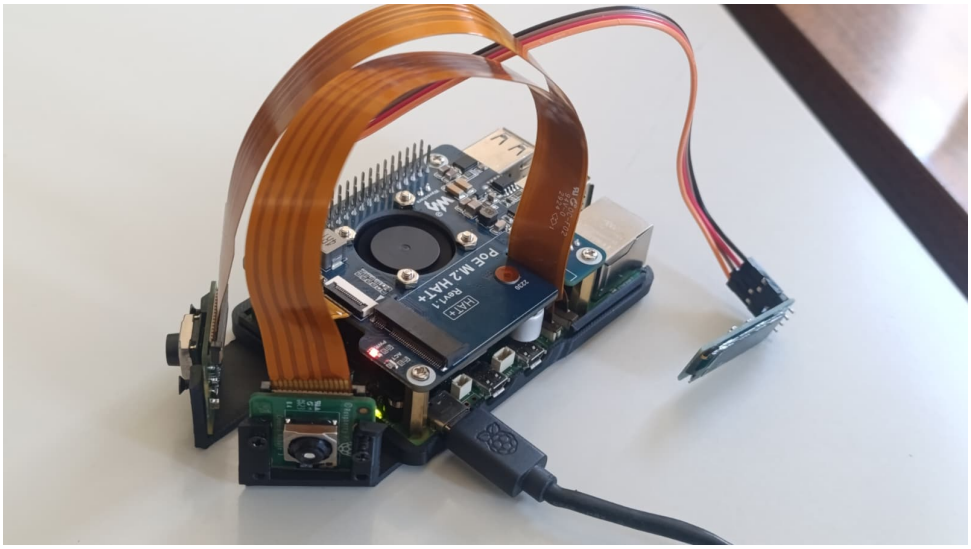


Figure 4.1: Prototype case holding the two selected cameras.

Following software testing with the initial prototype, the selected configuration was fixed in hardware. A custom physical enclosure was modeled using three dimensional computer aided design tools to securely hold both cameras at the exact orientation and tilt angles determined during the spatial simulation phase, as shown in Figure 4.2. The mechanical design of the enclosure

accommodates the Raspberry Pi 5 and the attached PoE HAT, allowing the complete system to be powered and connected through a single network cable. Thermal management was directly addressed in the enclosure structure by incorporating continuous ventilation slots aligned with the active fan and heat sink of the PoE HAT, preventing thermal throttling during repetitive neural network execution. The housing was manufactured using 3D printing with durable plastic filament to ensure structural integrity and environmental protection under outdoor conditions. Figure 4.3 shows the printed enclosure fully integrated and installed at the authorized mounting point B.

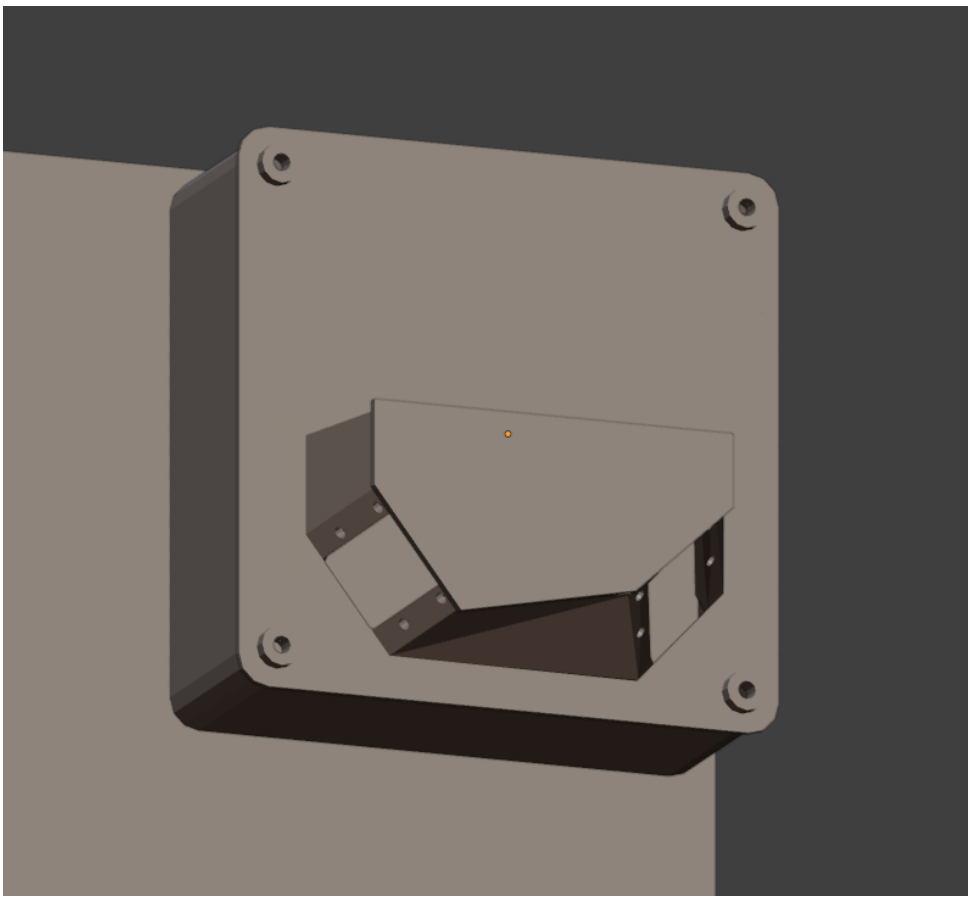


Figure 4.2: 3D model of the camera enclosure.

4.3 Counting Module

The counting module acts as the vision processing engine of the edge architecture, executing continuously during canteen service hours. To maintain reliable tracking while preserving board computational resources, the pipeline processes captured frames at 10 s intervals. This operational cadence provides a balance between capturing dynamic queue fluctuations and preventing unnecessary processor load or thermal accumulation on the single board computer. Captured visual data is processed strictly within volatile memory, ensuring that no raw frames or personally identifiable visual features are persisted to non volatile storage at any stage of execution.

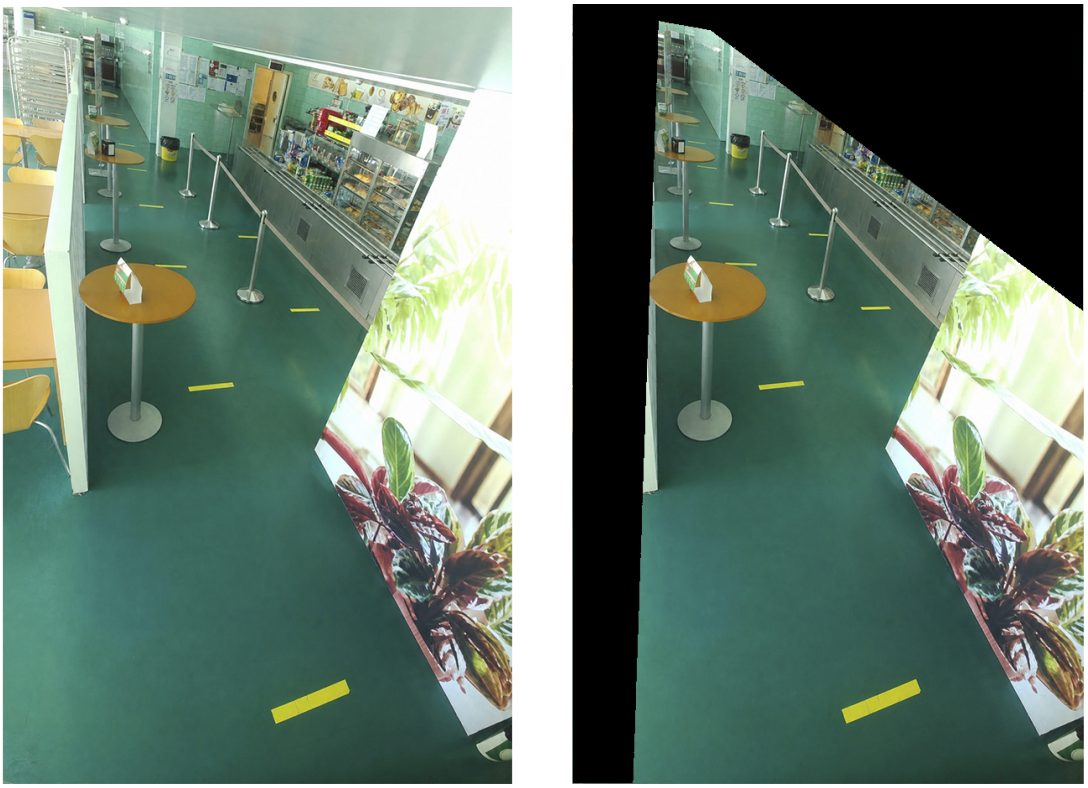


Figure 4.3: Printed camera enclosure installed at the mounting point.

Before passing visual frames to the neural network inference engine, a spatial filtering step is applied using a polygonal ROI. Physical placement constraints at the deployment site introduce adjacent pedestrian flows, including individuals walking along the main hallway and patrons standing in a parallel service line as described in Section 3.1. To prevent these non queue individuals from contaminating the occupancy estimate, the polygonal ROI masks out all image areas outside the designated waiting corridor, as represented in Figure 4.4.

To counteract the static artifacts observed after installation, a background calibration procedure was performed. During this calibration phase, multiple frames of the unoccupied corridor were processed to quantify residual background activations. The resulting baseline value is saved as a fixed calibration offset and subtracted from runtime counts, preventing static background features from biasing the final occupancy calculation.

To minimize unnecessary computation and reduce processing latency during off peak periods, the secondary camera operates conditionally. The counting engine continuously monitors the lower region of the primary camera view. When individual presence is detected within this trigger threshold, indicating that the queue has filled the primary corridor and is extending beyond the



(a) Full capture from camera (b) Applied ROI to camera view

Figure 4.4: Example captures from the mounting point.

corner, the system activates the secondary camera module. The frame from the secondary sensor is then processed, and its calculated occupancy is added to the primary count to yield total queue occupancy.

Upon completing frame inference, ROI filtering, and offset subtraction, the final occupancy count is written to a shared JSON file. To prevent race conditions with downstream estimation services that monitor this file, data writing is performed atomically through temporary file substitution. This file based communication interface decouples the visual processing module from the waiting time calculation engine, allowing inference to proceed at its independent operational rate while guaranteeing data integrity across the system architecture.

4.4 Estimation and Logging Module

The estimation and logging module operates as a dedicated background service managed by the operating system daemon, executing concurrently with the visual processing pipeline. The estimation service continuously monitors the shared file `status.json` for state updates published by the vision module. Upon detecting an atomic write operation, the service reads the newest count payload into memory. This file watcher mechanism ensures that the estimation pipeline reacts instantly to incoming visual measurements without introducing computational overhead or blocking the execution of the primary counting engine.

Raw visual counts captured at ten second intervals remain subject to short term fluctuations caused by detection noise, momentary body occlusions, and transient pedestrian movements. To mitigate this variability, the estimation module accumulates incoming counts into a non overlapping window of six consecutive measurements. Once six valid samples are collected, the module computes their arithmetic mean to establish a smoothed occupancy baseline representing a one minute operational window. The calibrated queueing estimator described in Methodology is then applied to this mean count, producing a single waiting time estimate approximately every minute. Equation 3.2 converts the aggregated occupancy into an expected wait duration, striking a balance between filtering high frequency counting noise and maintaining responsive updates for end users.

As each waiting time estimate is computed, the module constructs a structured JSON payload containing the mean occupancy, the calculated waiting time, a high resolution timestamp, and relevant system health indicators. This payload is transmitted immediately to the central application server using an Hypertext Transfer Protocol (HTTP) `POST` request directed to the endpoint described in Section 4.5. Network requests are handled synchronously with timeout constraints to prevent network latency from accumulating or disrupting subsequent estimation windows. If a transmission attempt fails due to temporary network unavailability or server errors, the event is flagged internally, allowing the local estimation loop to proceed without loss of operational continuity.

To ensure data resilience and support detailed offline analysis, the module maintains a persistent local record of all operational activity. Every individual count, aggregated window mean, generated waiting time estimate, and network transmission status is appended to a daily CSV log file stored on the edge device. The file writing routine employs thread safe append operations, preventing data corruption in the event of unexpected power disruptions. These daily logs serve as a local fallback archive, preserving complete operational history even during prolonged network outages.

At the conclusion of daily service hours, the module executes a structured shutdown protocol to finalize operational tasks. When a termination signal is issued by systemd or when an explicit closing status is read from `status.json`, the service closes the active CSV file stream and transmits the complete daily log file to the remote server using the dedicated upload endpoint. Following successful log transfer, the module dispatches a final termination payload containing a session closing flag. This payload notifies the central web server that service has ended for the day, prompting the dashboard to transition into an inactive state. Once the server acknowledges the final packet, the estimation process releases all allocated system resources and terminates gracefully.

4.5 Server and Public Dashboard

The server is a Flask application that receives, stores, and publishes the measurements produced on the edge device. Estimates arrive through an HTTP `POST` endpoint, which validates the required fields before accepting a packet. Each accepted measurement atomically updates a snapshot file holding the most recent state, following the same pattern as the inter-module interface on the edge

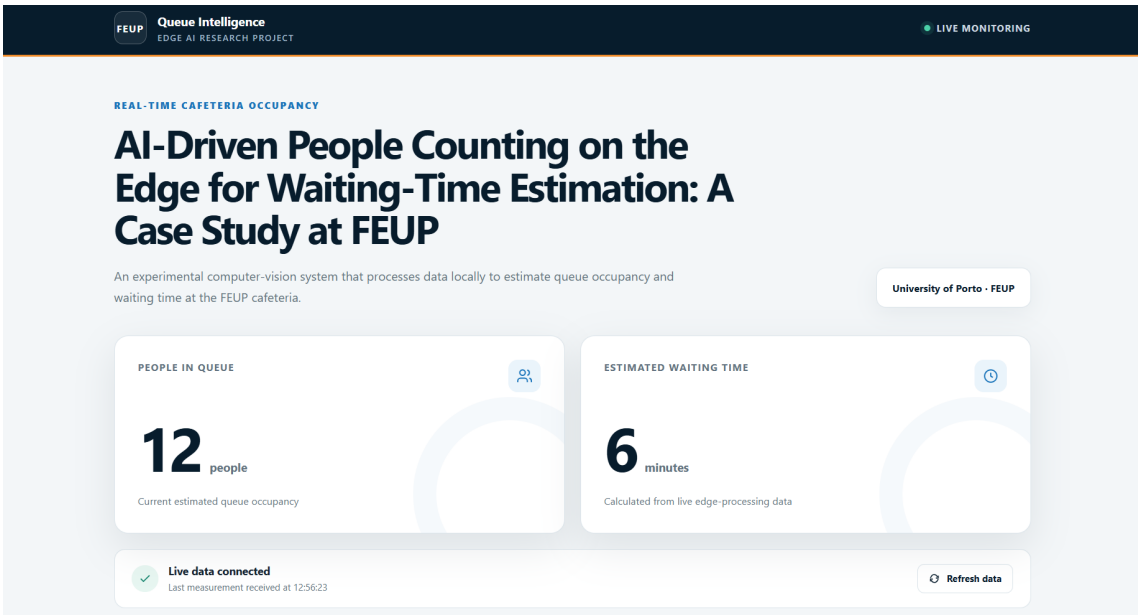


Figure 4.5: Public dashboard served by the system.

device. A second endpoint receives the daily CSV logs uploaded at shutdown, which constitute the accumulated record of the deployment. A read endpoint exposes the current snapshot to the dashboard, and all Application Programming Interface (API) responses carry no-cache headers, so that clients always retrieve the current state rather than a stale copy.

The dashboard, shown in Figure 4.5, presents the current number of people in the queue and the estimated waiting time. If no update has been received for five minutes, the values are replaced by an inactive-system indication, preventing outdated readings from being mistaken for current ones. The page also displays a privacy notice informing users that images are processed in real time and never stored, making the data-minimization principles of Section 2.8 visible at the point of use.

The machine hosting the server sits inside the university network and is reachable only through the institutional Virtual Private Network (VPN), which would restrict the dashboard to connected members. To make it publicly accessible, the server is exposed through a Cloudflare Tunnel: a persistent outbound connection from the host to Cloudflare's edge, which publishes the service at `https://filafeup.online` without requiring a public Internet Protocol (IP) address or open inbound ports.

4.6 System Integration

The complete system operates on a daily cycle aligned with the service's operating hours. Each module runs as a system service, started independently by a scheduled task at the opening time; since the modules communicate only through the shared JSON file, no startup ordering is required, and the estimation module simply waits for the first count to be written. From that point the system runs unattended: counts propagate through the file interface, estimates are transmitted as

they are produced, and the dashboard reflects the current state. When the operating hours end, the estimation module uploads the daily logs, sends the termination message, and both modules exit, leaving the device idle until the next opening.

Chapter 5

Results and Discussion

This chapter presents the results obtained with the deployed system. It begins with the accuracy of the counting against ground truth observed on site, followed by the behavior of the two-camera arrangement during operation and the performance of the device that hosts them. The waiting-time estimates are then compared with waits measured at the queue, and the data flow is examined against the privacy requirements established for the deployment. The chapter closes with a discussion of the main findings and of the limits within which they hold.

5.1 Counting Accuracy

Counting accuracy was evaluated against ground truth obtained through direct observation, as described in Section 3.7. The evaluation comprises 100 paired readings, with ground-truth occupancies ranging from 2 to 17 people. This range represents the low-to-moderate occupancies observed during the validation session, but it does not cover the highest estimates present in the longer operational log. Figure 5.1 plots the system counts against the ground truth, and Table 5.1 summarizes the error metrics overall and by occupancy band.

Table 5.1: Counting accuracy in deployment by ground-truth occupancy range.

Occupancy	**n**	**MAE**	**RMSE**	**NAE**	**Bias**
2–5	7	1.17	1.39	35.6%	+0.47
6–10	59	2.84	3.37	34.7%	+2.29
11–17	34	3.70	4.58	28.1%	+1.15
All	100	2.93	3.62	33.0%	+1.85

Across the complete sample, the system deviated from the ground truth by 2.93 people on average and achieved a Pearson correlation of $r = 0.678$. The positive mean bias of $+1.85$ people

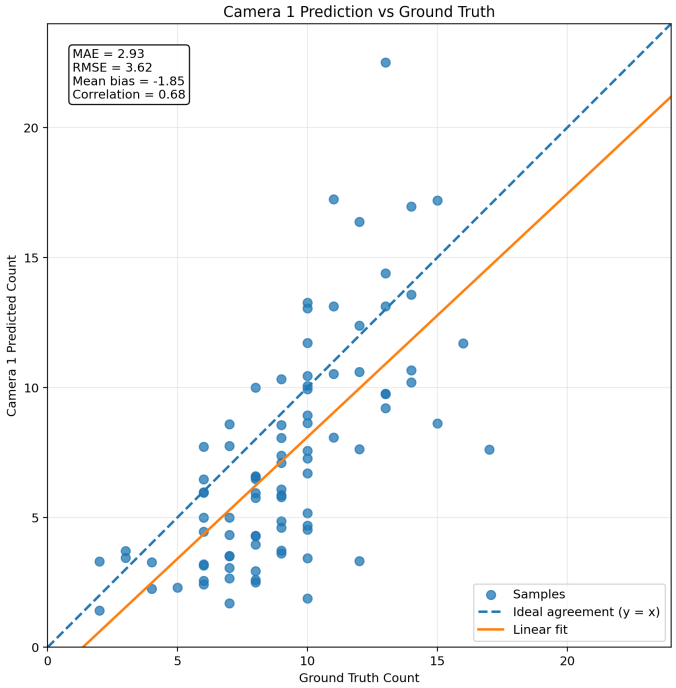


Figure 5.1: System count against manually observed ground truth.

indicates systematic overcounting. The model was trained on datasets of dense outdoor crowds and is applied here to a sparse indoor scene, which may account for part of this excess.

The absolute error increased with occupancy, from an MAE of 1.17 for occupancies of 2–5 people to 3.70 for occupancies of 11–17 people. The corresponding NAE decreased from 35.6% to 28.1%, indicating that the increase in absolute error was smaller than the increase in queue size. Field performance was nevertheless lower than the 10.9% NAE obtained by the same model on ShanghaiTech B in Table 3.1. The comparison should be interpreted cautiously because the benchmark dataset and the deployment differ in viewpoint, scene geometry, occlusion pattern, and occupancy distribution.

5.2 Operation and the Two-Camera Arrangement

The visual comparison between the simulated and installed views in Figure 4.4 shows that the planned camera orientation reproduced the intended coverage without requiring adjustment after installation. This validation is qualitative, since no geometric image-registration error was calculated, but it confirms that the three-dimensional reconstruction was sufficiently accurate to support camera and enclosure design.

Occupancy varied markedly between operating days, consistent with the variability identified during the field study. The logged period coincided with the end of the teaching term and may therefore not represent the usual peak demand at the site. Figure 5.2 shows the estimated occupancy on 26 May 2026, when the queue increased around noon and subsequently decreased as

demand subsided and the service processed the queue.

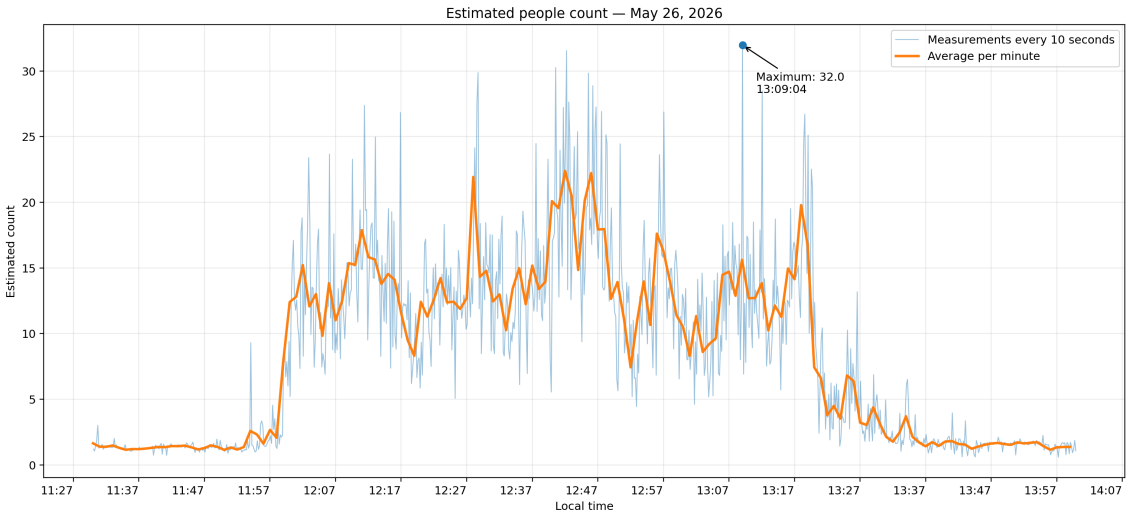


Figure 5.2: Estimated occupancy during the operating period of 26 May 2026.

The operational log covers 24 dated sessions between 25 May and 25 June 2026. After excluding 22 termination messages, it contains 3,251 one-minute estimates, equivalent to approximately 54.2 h of estimation windows. Eighteen sessions cover the complete 11:30–14:00 service period, while the remaining six are partial or contain an interruption.

The second camera was activated in 218 estimate records, corresponding to 6.7% of all estimates. Its limited activation is consistent with the secondary camera being used only when the queue extended into the additional segment. This conditional arrangement reduced unnecessary inference during most of the deployment. However, because the logged period contained relatively few sustained high-occupancy intervals, the benefit of the second camera under prolonged peak demand could not be fully characterized.

5.3 On-Device Performance

Table 5.2 summarizes the operational records. Of the 3,251 estimate transmissions, 3,146 received a response, corresponding to a delivery rate of 96.8%. The remaining 105 attempts comprised 55 name-resolution failures, 39 server side errors, and 11 read timeouts. The name-resolution failures formed a continuous outage on 5 June, whereas the other errors were mostly isolated. Because estimates continued to be written to the local log, communication failures did not stop counting or waiting-time estimation. For successfully delivered estimates, the time from the measurement timestamp to completion of processing and server acknowledgment averaged 3.04 s, with a median of 3 s, a 95th percentile of 6 s, and a maximum of 11 s. This delay is small relative to the one-minute estimation window, although the failed transmissions demonstrate that end-to-end availability depends on the network and server as well as on the embedded device. Figure 5.3

Table 5.2: Operational performance derived from the deployment log. Termination messages are excluded from the estimate and transmission totals.

Operational indicator	**Result**
Logged period	25 May–25 June 2026
Dated operating sessions	24
Complete service-window sessions	18
One-minute estimate records	3,251
Equivalent estimation-window time	54.2 h
Intervals at the nominal 60 s cadence	3,223 (99.9%)
Secondary camera activation records	218 (6.7%)
Successful transmissions	3,146 (96.8%)
Failed transmissions	105 (3.2%)
Mean successful-delivery latency	3.04 s
Median / 95th-percentile latency	3 s / 6 s
Maximum successful-delivery latency	11 s

shows the published result at the end of this path, on the class of device from which the queue is consulted before leaving for the canteen.

During daily operation of the deployed program, the process occupied approximately 772 MB of resident memory, while the recorded processor temperature ranged from 47.4 °C to 60.05 °C. No interruption attributable to memory exhaustion or excessive processor temperature was observed during the monitored runs.

5.4 Waiting-Time Accuracy

The vision stage showed systematic overcouFor the session shown in Figure 5.2, one-minute waiting time estimates were compared with waiting times measured on site. Table 5.3 summarizes the comparison. The estimates deviated from the measured waits by 0.93 min on average and showed a mean bias of −0.52 min, indicating systematic underestimation. This bias does not follow from the overcount reported in Section 5.1, since a higher occupancy produces a longer waiting-time estimate. However, the contribution of counting error and the contribution of the queueing approximation were not isolated experimentally and should not be treated as numerically equivalent.ting, while the waiting-time estimates fell below the measured waits. The experiment did not separate the effects of counting error from those of the queueing model.

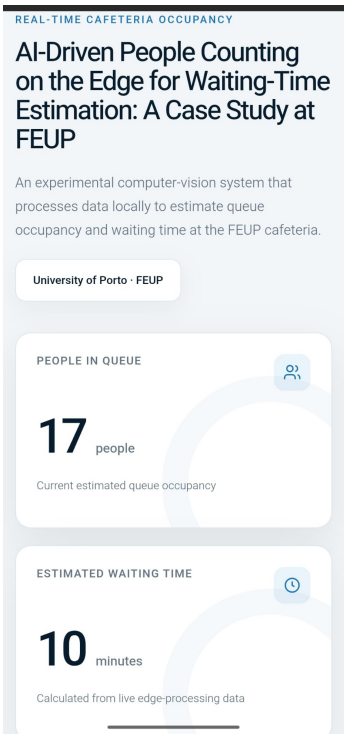


Figure 5.3: Published estimate as served to a mobile browser during operation.

Correlation-based metrics were not applied to the waiting time comparison. The measured waits available from a single session are too few for a correlation coefficient or a coefficient of determination to be meaningful, and their characterization over a larger and more varied set of measured waits is left for future work. Within the available sample, the absolute-error metrics indicate that the estimate approximated the magnitude of the wait well, with a MAE below one minute and a relative error near one fifth of the measured value.

5.5 Privacy in Operation

During operation, only numerical indicators left the device. The packets transmitted to the server and the local daily logs contained counts, waiting-time estimates, timestamps, transmission out-

Table 5.3: Waiting-time estimates against measured waits over one session.

Metric	**Result**
MAE	0.933 min
RMSE	1.104 min
Mean bias error	−0.517 min
MAPE	21.69%

comes, and system events; no images were transmitted or retained. Ground truth was collected through direct observation and stored only as timestamped numerical measurements. These observations show that the implemented data flow remained consistent with the privacy measures described in Section 3.6, without constituting an independent legal determination of GDPR compliance.

5.6 Discussion

The results support the technical feasibility of estimating queue occupancy and waiting time on a low-cost embedded device under the physical and computational constraints of the deployment site. The local pipeline produced 3,251 estimates over 24 dated sessions and maintained its nominal cadence in 99.9% of uninterrupted intervals. Representative resource measurements also remained within the available memory and thermal capacity of the Raspberry Pi 5.

Local reliability must nevertheless be distinguished from end-to-end availability. Although 96.8% of estimate transmissions were acknowledged by the server, 105 attempts failed, including a prolonged outage on 5 June. Local logging preserved the numerical results and estimation continued, but missed updates were not recovered by the dashboard. A retry queue or store-and-forward mechanism would therefore be needed to make publication as resilient as local processing.

The vision stage showed systematic overcounting, while the waiting-time estimates fell slightly below the measured waits. The experiment did not separate the effects of counting error from those of the queueing model. The camera-planning method produced the intended field of view without physical iteration, but its validation was qualitative.

The evaluation also has important scope limitations. Counting ground truth was collected in a single 50-minute session and covered occupancies from 2 to 17 people, while the longer operational log contained higher system estimates. Waiting-time accuracy was assessed during the same single session, and no user study evaluated whether the published information changed decisions or perceived waiting quality. In addition, the deployment occurred near the end of the teaching term, and the secondary camera was activated in only 6.7% of estimate records. Longer evaluation during sustained peak demand is therefore required to characterize high-density counting performance, the full contribution of the second camera, and the generalizability of the waiting-time results.

Chapter 6

Conclusion and Future Work

This dissertation seeks to address practical needs in daily life by instrumenting physical spaces to support informed decision making about routine activities, particularly queue waiting times. The motivation stems from the convergence of key technological trends. Advances in SBCs and ML have unlocked Edge AI, while sensing and data driven services are expanding across smart cities and smart campuses.

To operationalize this vision, a set of specific objectives was defined around the primary goal of developing an affordable, edge-based monitoring solution that estimates the waiting time of a queue and makes it easily accessible to its users. This purpose was pursued through four goals, covering the study of available technologies, the selection of hardware, the development of a privacy-compliant system, and its evaluation in a real scenario. Each of these goals was reached, demonstrating that a low-cost edge architecture can reliably transform visual observation into public information.

The first objective focused on mapping existing technologies for counting people from physical sensors, reviewing approaches scattered across the literature. As established in Chapter 2, prior research often relied either on non-visual sensors with spatial resolution limits, off-device processing that raises privacy concerns, or heavy neural networks suited especially to desktop hardware. By synthesizing techniques from computer vision, edge computing, and queueing theory, this work defined an operational gap for a privacy-preserving, edge-based solution. The review that resulted from this study, together with a comparison of the surveyed systems, stands as a contribution in itself and a starting point for future research in the area.

Furthermore, evaluating lightweight counting models, such as the optimized EBC-ZIP family and fine-tuned YOLO detectors, across multiple benchmark datasets directly on the edge device established a technical benchmark to guide future edge-vision implementations.

The second objective addressed the selection and physical planning of embedded hardware suited for monitoring a real service point under cost, autonomy, and installation constraints. The Raspberry Pi 5 was selected, offering a margin of processing capacity above the immediate needs of the task and easing development through its broad ecosystem, camera interfaces, and vast support. To overcome physical constraints and avoid trial and error on site, the canteen queueing area

was reconstructed in three dimensions using Blender. This simulation proved essential, guiding the choice of optics and informing the design of a custom 3D-printed enclosure that positioned sensors at the intended angles upon physical installation.

The third objective centered on integrating software and hardware into a modular system compliant with privacy constraints under the GDPR. Following the modular decomposition criteria of Parnas [55], the edge architecture was split into two independent services managed by the operating system that exchange data through an atomically updated status file. The local counts were translated into waiting times via an $M/M/1$ queueing model calibrated with an observed service rate of $\mu = 2.85$ people per minute. A two-camera arrangement was implemented to observe the L-shaped queue. However, due to physical mounting constraints at the authorized site, the cameras could not fully cover the entire line during peak extensions, a limitation that offers a clear direction for future hardware adjustments.

To guarantee privacy by design, captured frames were held strictly in volatile memory during EBC-ZIP inference and immediately discarded, ensuring that no imagery was ever stored or transmitted. The resulting estimates were transmitted to a Flask server and displayed on a public dashboard hosted at `https://filafeup.online` via a Cloudflare Tunnel. This architecture underwent formal appraisal by the Data Protection Unit of the University of Porto, confirming its alignment with institutional and regulatory privacy standards.

The fourth objective called for evaluating the deployed system in its real world application scenario against ground truth data observed on site. Over 24 dated operating sessions spanning 54.2 hours, the local pipeline generated 3,251 one-minute estimates and maintained its nominal cadence in 99.9% of uninterrupted intervals, operating within safe memory (772 MB) and thermal limits (47.4 °C to 60.05 °C). End-to-end delivery reached 96.8%, with 105 failed transmissions primarily caused by a temporary name-resolution outage. In field accuracy tests covering 100 paired observations, occupancy estimation yielded an MAE of 2.93 people and a mean bias of $+1.85$ people, reflecting systematic overcounting. The verified waiting-time estimates achieved an MAE of 0.933 minutes with a mean bias of -0.517 minutes. Although the evaluation was bounded by a single 50-minute ground truth session near the end of the academic term, where the secondary camera was active in 6.7% of logged intervals, the collected metrics confirm that the embedded deployment reliably sustained continuous operation within practical operational margins.

Future work can develop along four major areas. First, the faults still present in the system can be addressed. A persistent retry queue on the device would store failed transmissions and resend them once connectivity returns, so that a failure of any cause, such as the prolonged name-resolution outage observed on 5 June, would cost only timeliness and not data. A watchdog able to detect stalled modules, together with alerts built on the health indicators the device already transmits, would make prolonged inactivity or repeated communication failures visible as they happen, completing the unattended operation that the design pursues.

Then, the evaluation itself could be extended. Observation during regular university activities would expose the system to sustained peak demand, higher occupancies, and frequent activation of

the second camera, while additional measured waits would allow the statistical characterization of the waiting-time estimate that the present sample could not support. Exploring alternative camera mounting angles or wide-angle lenses could also address the sightline limitations identified at the current mounting point.

The system could also be further improved in what it offers. The fixed service rate could be replaced by a value updated from recent throughput observations, and alternative queueing formulations or data-driven estimators could be compared where the assumptions of the current model are not met. The dashboard, which today shows the current state, could grow into a tool for the service itself. Exposing the accumulated history, daily occupancy profiles, and expected waits by time of day would let the canteen anticipate demand and adjust its operation, and the same data could support forecasting for the users.

Finally, the architecture could be carried beyond this deployment. The 10-second capture interval leaves a wide throughput margin on the Raspberry Pi 5, so the pipeline is expected to fit boards of lower cost, where memory becomes the binding constraint. The project could also evolve into a more general solution, packaged so that it can be installed with little effort in other queues and shared spaces.

Ultimately, this dissertation demonstrates a working implementation of an end to end queue monitoring service, integrating elements that were previously addressed in isolation within the literature. It proves that a complete privacy preserving pipeline, from design to public data delivery, can operate effectively on affordable embedded hardware. By delivering a deployed, documented, and open source solution, this work establishes a solid technical foundation at FEUP for future smart campus initiatives.

References

[1] Hartmut Rosa. *Social Acceleration: A New Theory of Modernity*. New York: Columbia University Press, 2013.

[2] David H. Maister. "The Psychology of Waiting Lines". In: *The Service Encounter: Managing Employee/Customer Interaction in Service Businesses*. Ed. by John A. Czepiel, Michael R. Solomon, and Carol F. Surprenant. Lexington, MA: Lexington Books, 1985, pp. 113–123.

[3] Gordon E. Moore. "Cramming more components onto integrated circuits". In: *Electronics* 38.8 (1965), pp. 114–117.

[4] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444. DOI: 10.1038/nature14539.

[5] Jose Sanchez Gracias et al. "Smart Cities — A Structured Literature Review". In: *Smart Cities* 6.4 (2023), pp. 1719–1743. DOI: 10.3390/smartcities6040080.

[6] Luigi Atzori, Antonio Iera, and Giacomo Morabito. "The Internet of Things: A Survey". In: *Computer Networks* 54.15 (2010), pp. 2787–2805. DOI: 10.1016/j.comnet.2010.05.010.

[7] Zhi Zhou et al. "Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing". In: *Proceedings of the IEEE* 107.8 (2019), pp. 1738–1762. DOI: 10.1109/JPROC.2019.2918951.

[8] Weisong Shi et al. "Edge Computing: Vision and Challenges". In: *IEEE Internet of Things Journal* 3.5 (2016), pp. 637–646. DOI: 10.1109/JIOT.2016.2579198.

[9] Ken Polin et al. "The Making of Smart Campus: A Review and Conceptual Framework". In: *Buildings* 13.4 (Apr. 2023), p. 891. ISSN: 2075-5309. DOI: 10.3390/buildings13040891. URL: https://www.mdpi.com/2075-5309/13/4/891 (visited on 07/03/2026).

[10] He Li, Lu Yu, and Wu He. "The Impact of GDPR on Global Technology Development". In: *Journal of Global Information Technology Management* 22.1 (Jan. 2, 2019), pp. 1–6. ISSN: 1097-198X. DOI: 10.1080/1097198X.2019.1569186. URL: https://doi.org/10.1080/1097198X.2019.1569186 (visited on 07/03/2026).

[11] Ala Al-Fuqaha et al. "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications". In: *IEEE Communications Surveys & Tutorials* 17.4 (2015), pp. 2347–2376. DOI: 10.1109/COMST.2015.2444095.

[12] Jayavardhana Gubbi et al. "Internet of Things (IoT): A vision, architectural elements, and future directions". In: *Future Generation Computer Systems* 29.7 (2013), pp. 1645–1660. DOI: 10.1016/j.future.2013.01.010.

[13] Hafedh Chourabi et al. “Understanding Smart Cities: An Integrative Framework”. In: *Proceedings of the 45th Hawaii International Conference on System Sciences (HICSS)*. IEEE, 2012, pp. 2289–2297. DOI: 10.1109/HICSS.2012.615.

[14] Emine Zehra Akgün et al. “Breaking Barriers for Breaking Ground: A Categorisation of Public Sector Challenges to Smart City Project Implementation”. In: *Public Policy and Administration* (2024). DOI: 10.1177/09520767241263233.

[15] Loris Plasson-Simoni, Paul-Antoine Bisgambiglia, and Thierry Antoine-Santoni. “Towards the integration of ambient intelligence in smart buildings: Definitions and propositions”. In: *Journal of Ambient Intelligence and Smart Environments* 18.1 (2025). Published online 30 October 2025; issue published February 2026, pp. 5–23. DOI: 10.1177/18761364251389165.

[16] Pascal Neis, Dominik Warch, and Max Hoppe. “Testing and Evaluation of Low-Cost Sensors for Developing Open Smart Campus Systems Based on IoT”. In: *Sensors* 23.20 (2023), p. 8652. DOI: 10.3390/s23208652.

[17] Lorenzo Monti et al. “Edge-Based Transfer Learning for Classroom Occupancy Detection in a Smart Campus Context”. In: *Sensors* 22.10 (2022), p. 3692. DOI: 10.3390/s22103692.

[18] Yitao Chen et al. *LiDAR-Based Outdoor Crowd Management for Smart Campus on the Edge*. Preprint. Nov. 29, 2023. DOI: 10.48550/arXiv.2311.18077. arXiv: 2311.18077 [cs.CV]. URL: https://arxiv.org/abs/2311.18077 (visited on 07/03/2026).

[19] Peter Marwedel. *Embedded System Design: Embedded Systems, Foundations of Cyber-Physical Systems, and the Internet of Things*. 4th ed. Springer, 2021. DOI: 10.1007/978-3-030-60910-8.

[20] Andrew G. Howard et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: 1704.04861.

[21] Song Han, Huizi Mao, and William J. Dally. “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding”. In: *4th International Conference on Learning Representations (ICLR)*. 2016. arXiv: 1510.00149.

[22] Hui Ni and The ncnn contributors. *ncnn*. original-date: 2017-06-30T10:55:37Z. June 2017. URL: https://github.com/Tencent/ncnn (visited on 07/07/2026).

[23] Azad Shokrollahi et al. “Passive Infrared Sensor-Based Occupancy Monitoring in Smart Buildings: A Review of Methodologies and Machine Learning Approaches”. In: *Sensors* 24.5 (2024), p. 1533. DOI: 10.3390/s24051533.

[24] Aleksander Radovan et al. “A Review of Passenger Counting in Public Transport Concepts with Solution Proposal Based on Image Processing and Machine Learning”. In: *Eng* 5.4 (2024), pp. 3284–3315. DOI: 10.3390/eng5040172.

[25] F. H. Wahl, M. Milenkovic, and O. D. Amft. “A Distributed PIR-Based Approach for Estimating People Count in Office Environments”. In: *2012 IEEE 15th International Conference on Computational Science and Engineering*. IEEE, 2012, pp. 640–647. DOI: 10.1109/ICCSE.2012.92.

[26] A. Soumya, C. Krishna Mohan, and Linga Reddy Cenkeramaddi. “Recent Advances in mmWave-Radar-Based Sensing, Its Applications, and Machine Learning Techniques: A Review”. In: *Sensors* 23.21 (2023), p. 8901. DOI: 10.3390/s23218901.

[27] Richard Szeliski. *Computer Vision: Algorithms and Applications*. 2nd ed. Cham: Springer, 2022. DOI: 10.1007/978-3-030-34372-9.

[28] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2nd ed. Cambridge: Cambridge University Press, 2004. ISBN: 978-0-521-54051-3.

[29] Vasu Parameswaran, Vinay Shet, and Visvanathan Ramesh. "Design and Validation of a System for People Queue Statistics Estimation". In: *Video Analytics for Business Intelligence*. Ed. by Caifeng Shan et al. Vol. 409. Studies in Computational Intelligence. Berlin, Heidelberg: Springer, 2012, pp. 355–373. DOI: 10.1007/978-3-642-28598-1_11.

[30] Jia Hong Yin, Sergio A. Velastin, and Anthony C. Davies. "Image Processing Techniques for Crowd Density Estimation Using a Reference Image". In: *Recent Developments in Computer Vision*. Ed. by Stan Z. Li et al. Berlin and Heidelberg: Springer, 1996, pp. 489–498. ISBN: 978-3-540-49448-5. DOI: 10.1007/3-540-60793-5_102. URL: https://doi.org/10.1007/3-540-60793-5_102 (visited on 07/03/2026).

[31] Vishwanath A. Sindagi and Vishal M. Patel. "A Survey of Recent Advances in CNN-Based Single Image Crowd Counting and Density Estimation". In: *Pattern Recognition Letters* 107 (May 1, 2018), pp. 3–16. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2017.07.007. URL: https://doi.org/10.1016/j.patrec.2017.07.007 (visited on 07/03/2026).

[32] Muhammad Asif Khan, Hamid Menouar, and Ridha Hamila. "Revisiting Crowd Counting: State-of-the-Art, Trends, and Future Perspectives". In: *Image and Vision Computing* 129 (Jan. 2023), p. 104597. ISSN: 0262-8856. DOI: 10.1016/j.imavis.2022.104597. URL: https://doi.org/10.1016/j.imavis.2022.104597 (visited on 07/03/2026).

[33] Yingying Zhang et al. "Single-Image Crowd Counting via Multi-Column Convolutional Neural Network". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 589–597. DOI: 10.1109/CVPR.2016.70. URL: https://doi.org/10.1109/CVPR.2016.70 (visited on 07/03/2026).

[34] Yuhong Li, Xiaofan Zhang, and Deming Chen. "CSRNet: Dilated Convolutional Neural Networks for Understanding the Highly Congested Scenes". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018, pp. 1091–1100. DOI: 10.1109/CVPR.2018.00120.

[35] Liang Liu et al. "Counting Objects by Blockwise Classification". In: *IEEE Transactions on Circuits and Systems for Video Technology* 30.10 (Oct. 2020), pp. 3513–3527. ISSN: 1558-2205. DOI: 10.1109/TCSVT.2019.2942970. URL: https://doi.org/10.1109/TCSVT.2019.2942970 (visited on 07/03/2026).

[36] Yiming Ma, Victor Sanchez, and Tanaya Guha. "CLIP-EBC: CLIP Can Count Accurately through Enhanced Blockwise Classification". In: *2025 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2025, pp. 1–6. DOI: 10.1109/ICME59968.2025.11209839. URL: https://doi.org/10.1109/ICME59968.2025.11209839 (visited on 07/03/2026).

[37] Yiming Ma, Victor Sanchez, and Tanaya Guha. *EBC-ZIP: Improving Blockwise Crowd Counting with Zero-Inflated Poisson Regression*. Preprint. June 24, 2025. DOI: 10.48550/arXiv.2506.19955. arXiv: 2506.19955 [cs.CV]. URL: https://arxiv.org/abs/2506.19955 (visited on 07/03/2026).

[38] G. Vasavi et al. “People Counting Based on YOLO”. In: *2023 4th IEEE Global Conference for Advancement in Technology (GCAT)*. IEEE, Oct. 2023, pp. 1–6. DOI: 10.1109/GCAT59970.2023.10353287. URL: https://doi.org/10.1109/GCAT59970.2023.10353287 (visited on 07/03/2026).

[39] Raghavendra Vadavadagi et al. “Head Counting in Crowded Scenes Using YOLOv10: A Deep Learning Approach”. In: *Proceedings of the 3rd International Conference on Futuristic Technology*. Pune, India: SCITEPRESS – Science and Technology Publications, 2025, pp. 611–618. ISBN: 978-989-758-763-4. DOI: 10.5220/0013633400004664. URL: https://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0013633400004664 (visited on 07/03/2026).

[40] Ruchika, Ravindra Kumar Purwar, and Shailesh Verma. “Analytical Study of YOLO and Its Various Versions in Crowd Counting”. In: *Intelligent Data Communication Technologies and Internet of Things*. Ed. by D. Jude Hemanth, Danilo Pelusi, and Chandrasekar Vuppalapati. Singapore: Springer Nature, 2022, pp. 975–989. ISBN: 978-981-16-7610-9. DOI: 10.1007/978-981-16-7610-9_71. URL: https://doi.org/10.1007/978-981-16-7610-9_71 (visited on 07/03/2026).

[41] Mehmet Şirin Gündüz and Gültekin Işık. “A New YOLO-Based Method for Real-Time Crowd Detection from Video and Performance Analysis of YOLO Models”. In: *Journal of Real-Time Image Processing* 20.1 (Jan. 30, 2023), p. 5. ISSN: 1861-8219. DOI: 10.1007/s11554-023-01276-w. URL: https://doi.org/10.1007/s11554-023-01276-w (visited on 07/03/2026).

[42] Aseem Rajvanshi and Dhanya Pramod. “Queue Time Estimation in Checkout Counters Using Computer Vision and Deep Neural Network”. In: *Journal of Physics: Conference Series* 1964.6 (2021), p. 062108. DOI: 10.1088/1742-6596/1964/6/062108.

[43] Yan Wang et al. “Measuring Human Queues Using WiFi Signals”. In: *Proceedings of the 19th Annual International Conference on Mobile Computing and Networking*. MobiCom ’13. New York, NY, USA: Association for Computing Machinery, 2013, pp. 235–238. ISBN: 978-1-4503-1999-7. DOI: 10.1145/2500423.2504584.

[44] Yan Wang et al. “Tracking Human Queues Using Single-Point Signal Monitoring”. In: *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*. MobiSys ’14. New York, NY, USA: Association for Computing Machinery, 2014, pp. 42–54. ISBN: 978-1-4503-2793-0. DOI: 10.1145/2594368.2594382.

[45] Jiangtao Wang et al. “Real-Time and Generic Queue Time Estimation Based on Mobile Crowdsensing”. In: *Frontiers of Computer Science* 11.1 (2017), pp. 49–60. DOI: 10.1007/s11704-016-5553-z.

[46] John D. C. Little. “A Proof for the Queuing Formula: $L = \lambda W$”. In: *Operations Research* 9.3 (1961), pp. 383–387. DOI: 10.1287/opre.9.3.383.

[47] Donald Gross et al. *Fundamentals of Queueing Theory*. 4th ed. Hoboken, NJ: Wiley, 2008. DOI: 10.1002/9781118625651.

[48] Ivo Adan and Jacques Resing. *Queueing Theory*. Eindhoven: Eindhoven University of Technology, 2002. URL: https://cse.sc.edu/~mgv/csce317sp15/adan_queueing2002.pdf (visited on 07/03/2026).

[49] *Regulation (EU) 2016/679 of the European Parliament and of the Council. Of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/EC*. Apr. 27, 2016. URL: https://eur-lex.europa.eu/eli/reg/2016/679/oj (visited on 07/03/2026).

[50] Arun Joseph and Vinod Ganapathy. "SoK: Evaluation of Methods for Privacy Preserving Edge Video Analytics". In: *Information Systems Security*. Cham: Springer Nature Switzerland, 2025, pp. 389–410. DOI: 10.1007/978-3-032-13714-2_24. URL: https://doi.org/10.1007/978-3-032-13714-2_24 (visited on 07/03/2026).

[51] Jean-François Determe et al. "Monitoring Large Crowds With WiFi: A Privacy-Preserving Approach". In: *IEEE Systems Journal* 16.2 (2022), pp. 2148–2159. DOI: 10.1109/JSYST.2021.3139756.

[52] Mertcan Cokbas, Prakash Ishwar, and Janusz Konrad. "Low-Resolution Overhead Thermal Tripwire for Occupancy Estimation". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2020, pp. 88–89. DOI: 10.1109/CVPRW50498.2020.00052.

[53] Ishrath Ahamed et al. *Real-Time AI-Driven People Tracking and Counting Using Overhead Cameras*. Preprint. Nov. 15, 2024. DOI: 10.48550/arXiv.2411.10072. arXiv: 2411.10072 [cs.CV]. URL: https://arxiv.org/abs/2411.10072 (visited on 07/22/2026).

[54] Chen Xie et al. "Efficient Deep Learning Models for Privacy-preserving People Counting on Low-resolution Infrared Arrays". In: *IEEE Internet of Things Journal* 10.14 (2023), pp. 12833–12844. DOI: 10.1109/JIOT.2023.3263290.

[55] D. L. Parnas. "On the Criteria to Be Used in Decomposing Systems into Modules". In: *Communications of the ACM* 15.12 (1972), pp. 1053–1058. DOI: 10.1145/361598.361623.

[56] Vinícius Caparróz Carvalho. *fila_FEUP_edge_device: Edge device software for privacy-preserving queue monitoring*. 2026. URL: https://github.com/MasterGoodbytes/fila_FEUP_edge_device (visited on 07/22/2026).

[57] Vinícius Caparróz Carvalho. *fila_FEUP_dashboard: Server and public dashboard for queue monitoring*. 2026. URL: https://github.com/MasterGoodbytes/fila_FEUP_dashboard (visited on 07/22/2026).

[58] The systemd Project. *systemd.service — Service unit configuration*. Linux manual pages, section 5. 2024. URL: https://man7.org/linux/man-pages/man5/systemd.service.5.html (visited on 07/08/2026).

[59] Ultralytics. *Configuration — YOLO Docs*. 2026. URL: https://docs.ultralytics.com/usage/cfg/ (visited on 07/08/2026).

[60] Ao Wang et al. "YOLOv10: Real-Time End-to-End Object Detection". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2024.

[61] Ekaba Bisong. "Google Colaboratory". In: *Building Machine Learning and Deep Learning Models on Google Cloud Platform*. Berkeley, CA: Apress, 2019, pp. 59–64. DOI: 10.1007/978-1-4842-4470-8_7.

[62] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. Amsterdam, 2025. URL: https://www.blender.org.