Journal of
Sensor and Actuator Networks

Article

MeSmart-Pro: Advanced Processing at the Edge for Smart Urban Monitoring and Reconfigurable Services

Antonino Galletta [1], Armando Ruggeri [1], Maria Fazio [1], Gianluca Dini [2,*] and Massimo Villari [1]

1 MIFT Department, University of Messina, 98166 Messina, Italy; angalletta@unime.it (A.G.); armruggeri@unime.it (A.R.); mfazio@unime.it (M.F.); mvillari@unime.it (M.V.)
2 Department of Information Engineering, University of Pisa, 56100 Pisa, Italy
* Correspondence: gianluca.dini@unipi.it

Received: 13 August 2020; Accepted: 22 November 2020; Published: 4 December 2020

Abstract: With reference to the MeSmart project, the Municipality of Messina is making a great investments to deploy several types of cameras and digital devices across the city for carrying out different tasks related to mobility management, such as traffic flow monitoring, number plate recognition, video surveillance etc. To this aim, exploiting specific devices for each task increases infrastructure and management costs, reducing flexibility. On the contrary, using general-purpose devices customized to accomplish multiple tasks at the same time can be a more efficient solution. Another important approach that can improve the efficiency of mobility services is moving computation tasks at the Edge of the managed system instead of in remote centralized serves, so reducing delays in event detection and processing and making the system more scalable. In this paper, we investigate the adoption of Edge devices connected to high-resolution cameras to create a general-purpose solution for performing different tasks. In particular, we use the Function as a Service (FaaS) paradigm to easily configure the Edge device and set up new services. The key results of our work is deploying versatile, scalable and adaptable systems able to respond to smart city's needs, even if such needs change over time. We tested the proposed solution setting up a vehicle counting solution based on OpenCV, and automatically deploying necessary functions into an Edge device. From experimental results, it results that computing performance at the Edge overtakes the performance of a device specifically designed for vehicle counting under certain conditions and thanks to our reconfigurable functions.

Keywords: edge computing; virtual device; OpenCV; traffic flow monitoring; FaaS

1. Introduction

Smart Cities represent a new way to live the urban environment, involving social and technological aspects. In this context, systems based on IoT (Internet of Things) play an important role to allows citizens to interact with the environment and to benefit of advanced services, such as video surveillance, intelligent traffic lightning, air quality sensing and noise estimation, to obtain intelligent means of transport and autonomous vehicles management [1]. From a technological point of view, using sensors and actuators to automatize services has become really strategic, but managing, configuring and optimizing the digital infrastructures to adapt their behavior to the specific needs of the context is a big challenge. The Municipality of Messina is working hard to make Massina a smart city, in order to improve the quality of life of its citizens and the quality of experience in accessing public services.

Messina is located in Sicily, an Italian island in the Mediterranean area of Europe. It can be considered as the "door" of Sicily because only a small portion of the sea, about 3 km long (Strait of Messina) separates Messina (and the rest of Sicily), from the Italian peninsula. The port of Messina is in a strategic position in the center of the city and acts as a multimodal hub for the metropolitan/regional network for handling freight, transport passengers from and to the rest of Italy. According to the most recent statistics, in 2019 more than 10 millions of passengers travelled through the Strait of Messina [2,3]. Messina is also a tourist destination with about 400,000 cruise passengers in its port per year, and it is one of the 15 biggest commercial ports in Italy with about 18 million tonnes of goods per year. The intense activity of multimodal transportation systems in the city center causes many problems to urban mobility. For that, the Municipality of Messina is investigating innovative solutions for mobility management, and this paper provides a contribution to its current activity in this field.

The Municipality of Messina is making a significant investment on transport services, buying several IP high-resolution cameras for video surveillance, Traficams for traffic flow monitoring, cameras for vehicle plate recognition and so on. However, buying specific devices for each service implies a high costs, both in terms of equipment purchase and in the management of a highly heterogeneous infrastructure. Furthermore, these devices may present several drawbacks. For instance, Traficams are quite expensive (the cost of each camera is about 2.000€) and are not able to accomplish multiple tasks. The resolution is quite low, therefore, at a certain time of the day, vehicle detection works poorly. On the contrary, using general-purpose devices and customizing them to accomplish multiple tasks can be a more efficient and cheap approach. Moving computation tasks at the Edge of the system instead of in remote centralized serves can make the system more scalable, reducing both delays in event detection and bandwidth usage. Our idea is to exploit general-purpose devices (e.g., high-resolution cameras) and employ Edge devices (e.g., Raspberry Pi 4 whose cost is only 70€) to accomplish different user-defined tasks such as person counting, object identification, vehicle number plate recognition, traffic monitoring etc.

From a technical point of view, we propose a system based on the Function as a Service (FaaS) computational paradigm. FaaS allows us to define several minimal functions (e.g., vehicle counting or vehicle number plate recognition) and to run one or more instances of them on the same device at the same time in order to implement scalable applications/services. In this paper, we present a vehicle counting system we developed to be run on a Raspberry Pi mod 4 equipped with a USB webcam with 1920×1080 resolution, and we compare the performance of the proposed approach with the performance of a Traficam device. Vehicle counting systems have been addressed in other works such as [4–7] however, to the best of our knowledge, it is the first time that FaaS is adopted on Edge devices for such purposes. We made different types of analyses to test both the feasibility of our Edge-based vehicle counting system and the applicability of these devices for heterogeneous purposes. Together with the vehicle counting system, we ran other concurrent services performing different tasks, such as OCR (Optical Character Recognition) for vehicle plate recognition, to set up a useful service for mobility management, and we also considered several connected cameras.

The contribution of this paper can be summarised as follows:

- investigating how to set-up of a large scale general-purpose and distributed environment able to deploy on-demand services;
- proposing a vehicle counting system based on the FaaS paradigm and Edge computing to setup concurrent services;
- evaluating the performance of our system based on general-purpose devices also in terms of scalability whenever multiple services run simultaneously;
- discussing how our approach can provide real advantages in comparison with solutions based on specific monitoring devices.

The rest of the paper is organised as follows: the current State of the Art is presented in Section 2, the motivation at the basis of this work are explained in Section 3, a brief discussion about the background is presented in Section 4, design and implementation details are presented in Sections 5 and 6 respectively, an evaluation of system performance is presented in Section 7 and, finally, Section 8 provides our conclusions.

2. State of the Art

In this section, we report a review of related works in the context of Edge Computing and Smart Cities, and Edge Computing and Video Analysis, with particular attention to safety and traffic management goals. Some works are available in the literature on these topics, however only a few deal with the capabilities of dynamically changing the computational behavior at the edge. Some of these works try to develop lightweight algorithms and computational load. Moreover, edge computing is often considered as an extension of cloud computations with limited resources (e.g., see [8–11]). Our work is different because we aim to adapt edge computing and load to the purpose of executing conventional algorithms on the cloud.

2.1. Smart Cities and Traffic Monitoring

A real-time traffic monitoring using IoT-based is discussed in [4]. Ultrasonic sensors are used to detect vehicle traffic levels at the lanes and to send them to the server for analysis. A similar work is presented in [5]. The authors, by using Zigbee sensors, send the traffic movements to a control center. A traffic signal automation system is presented in [6]. The proposed system, by using IR sensors is able to calculate the traffic density and to switch the traffic lights accordingly. A very interesting Edge-based video analytic system for traffic monitoring is presented in [7]. The proposed systems are very interesting, however, relying on the server for the analysis, in case of disconnection, they, contrary to our system, are not able to perform the analysis. Furthermore, our system is based on FaaS that allows it to be easily reconfigured to accomplish multiple tasks. The works presented above are designed and implemented for a specific goal, therefore, to add or remove functionalities imply the redesign of the whole system.

2.2. Edge Computing and Smart Cities

An interesting survey has been carried out in Edge Computing Enabled Smart Cities: A Comprehensive Survey [12], authors highlight the role of edge computing to realize the vision of smart cities analyzing the evolution of edge computing paradigms, with the objective to classify the literature by devising a comprehensive and meticulous taxonomy. They identify and discuss key requirements, and enumerate recently reported synergies of edge computing enabled smart cities. Finally, several indispensable open challenges along with their causes and guidelines are discussed, serving as future research directions. Intelligent Offloading for Collaborative Smart City Services in Edge Computing [13] claims the weakness of long service response time and low QoS (Quality of Service) in scenarios with clouds. The authors remark that edge computing is nowadays integrated with the smart city to promote the inherent shortcomings of terminals in cities, to this they designed an intelligent offloading method for collaborative smart city services, named IOM. They try to achieve a trade-off among minimizing service response time, optimizing energy consumption and maintaining load balance while guaranteeing privacy preservation during service offloading. A comprehensive and good analysis with mathematical models has been done.

2.3. Edge Computing and Video Analysis

The collaborative cloud and edge for object tracking are presented in [8], where ML algorithms are described in the approach of a partial processing of video acquisition on the edge, as compared with higher complexity of processing on the cloud. More results using different tracking strategies are reported. However, this work uses edge computing for limited purposes.

Edge networks created among devices are used in cooperative caching and video characteristics in [14] where opportunistic algorithms for sharing chunks of videos are considered. Here, we are much more in the domain of video analysis in terms of energy consumption during the sharing. At the Edge, Device-to-Device (D2D) is powered from batteries, since their simulation models try to understand better policies in different scenarios.

The Adaptive Wireless Video Streaming Based on Edge Computing cited in [15] is aligned with our approach that is the use of the emerging edge computing paradigm by deploying edge transcoding servers close to base stations. Their idea is to adopt the Dynamic Adaptive Streaming over HTTP (DASH) for edge transcoding where servers cooperate with the base station to transcode videos at a finer granularity according to the obtained users' channel conditions, which smartly adjusts the transcoding strategy to tackle time-varying wireless channels. Varying at the edge computations is also our goal in this work.

2.4. Edge Computing and Video Analysis for Safety

A Lightweight Deep Learning based Intelligent Edge Surveillance system is presented from the authors in [10]. They report a lightweight neural network at the edge node, in particular, they use a modified version of MobileNetV2-SSD for the IIoT applications, which combine edge computing and cloud computing. It is interesting to see how a lightweight computation is achieved at the edge nodes. It can represent our future work in this perspective for intelligent behaviors at the edge.

The deep learning-based couple-like cooperative computing method for the IoT-based Intelligent Surveillance Systems is presented in [16] and uses an intelligent approach at the edge. Here, the authors leverage the NVIDIA-embedded development board, Jetson TX2 (NVIDIA Corporation, Santa Clara, CA, USA) This board is intended for artificial intelligence based on the NVIDIA Pascal system, whereas its compact size and powerful computing capabilities make it simple to integrate into various edge devices. This work detects helmet-wearing and confirms the workers' identities. In the future, they want to combine this method with some others to achieve cross-camera detection.

The research presented in [17] is in line with our current outcomes of the presented paper about the emergence of edge computing being highly promising in video pre-processing with an edge camera. Authors claim that a video surveillance system is a killer application for edge computing. They propose an edge computing-enabled video usefulness system aimed at large-scale video surveillance systems. They provide early failure detection and bandwidth improvement. Their model can locate a failure and send it to end-users on the fly. Their experimental results demonstrate the approaches in the video usefulness model accurately detect failures that were collected from a video surveillance system with approximately 4000 cameras. This represents a useful case study for at edge computations.

Finally, in [9] authors recognize that many smart video surveillance approaches are proposed for object detection and tracking by using Artificial Intelligence (AI) and Machine Learning (ML) algorithms. However, it is still hard to migrate those computing and data-intensive tasks from cloud to edge due to the high computational requirement. Their hybrid Lightweight Tracking Algorithm to Enable Smart Surveillance as an edge Service describes intelligent surveillance as an edge service by proposing a hybrid lightweight tracking algorithm named Kerman (Kernelized Kalman filter) that is a decision tree-based hybrid Kernelized Correlation Filter (KCF) algorithm proposed for human object tracking, which is coupled with a lightweight Convolutional Neural Network (L-CNN) for high performance. Their proposed Kerman algorithm has been implemented on a couple of single board computers (SBC) as edge devices and validated using real-world surveillance video streams. The experimental results are promising, showing that the Kerman algorithm is able to track the object of interest with decent accuracy at a resource consumption affordable by edge devices.

2.5. Edge Computing and Video Analysis for Traffic

Interesting work has been conducted in [18], where the real-time traffic light control system based on background updating and where the detection is performed at the edge, similarly to what we describe in the following sections of this paper. However, they focus the excellent analysis on how to adjust the traffic light based on traffic volume, no consideration is given to the computation capabilities of the edge node, as we made.

Similarly to the above-presented work, in [19] the video-Based Traffic Flow Monitoring algorithm for single-phase position at the intersection is treated. Their innovation is based on new algorithms to obtain background differences to guarantee edge information and extract a complete moving target, using Canny operator-based edge detection to preserve weak edges and taking advantage of median filter to remove noise obtaining a clearer processed image. No consideration is performed like in our case about the computation capabilities.

The difficulties of our analysis reside in the different setup necessary to put in place with all weather and illumination conditions. The work in [20] is interesting because opportunely address these issues driving video mining.

2.6. Edge Computing and Function as a Service

The benefits of a serverless architecture are discussed in [21]. The authors propose a FaaS platform for Edge computing and propose several application scenarios to test the applicability. The authors in [22] discussed the existing FaaS technologies and highlighted the use of the WebAssembly paradigm to enable the computation at the Edge. A federation of serverless Edge-based systems is proposed in [23]. In particular, the authors proposed a prototype for oil extraction and analyzed the performance of the proposed system. Qualitative and quantitative analyses of four open-source FaaS frameworks are discussed in [24]. In order to compare these frameworks, the authors deployed several services that push environmental parameters to a serverless edge-based platform. The results showed that all frameworks have similar results.

3. Motivation

Edge computing is becoming a new challenging approach to provide advances in distributed and scalable systems. Providing services closer to end-users is a great opportunity, especially when low network latency is a mandatory requirement and connections to the remote computing systems (e.g., cloud datacenters [11]) are not always available. Intelligent systems at the edge help to foresee new opportunities also in smart cities, where the efficient management of computation and storage resources becomes essential for improving the quality of life of citizens.

3.1. The Mesmart Project at a Glance

Mesmart is a project of the Municipality of Messina funded by the Italian National Operational Program (PON) for Metropolitan City. It integrates actions of the Digital Agenda that mainly concern government and tourism management with particular attention to the issues related to data and processes for the digitalization of services. The Mesmart project is looking at providing and improving new digital services aimed at municipality employees and citizens. Several actions have been planned, and some of them deal with Video Surveillance and environmental monitoring, including the observation of air quality (e.g., temperature, humidity, luminosity), air pollution (using PM2 and PM 10 sensors), noise pollution, electromagnetic pollution, and waste management, water monitoring and traffic control. The Messina Municipality is making a meaningful investment by buying specific equipment for each action, such as IP cameras with high resolution that will be located in many parts of the city for video surveillance, traficams (which are cameras aimed at vehicle presence detection) for traffic flow monitoring, certified electromagnetic and noise sensors, etc. To provide advanced services for the smart city, usually, raw data and videos are sent to remote cloud servers to be processed [25]. However, this approach for service provisioning presents some drawbacks, such as: (1) high costs for equipment purchase and for the management of a highly heterogeneous infrastructure; (2) low flexibility in updating deployed services according to the new requirements that could arise during the time; (3) delays in service response due to network issue between a local system, where the service is provided to citizens and cloud datacenters, where the service is processed. The work presented here aims to overcome the above issues proposing an Edge-oriented approach, In particular, we implemented a new solution in the context of video and audio processing for urban mobility management, where the configuration and setup of services will change accordingly to the needs of Messina Municipal Police which is in charge of constant monitoring of the urban environment.

3.2. Video and Audio Processing at the Edge: An Example of a Virtual Device

The key idea of our work is to use cameras for multiple purposes, such as safety and security. The Police Department must have continuous access to live videos to constantly supervise the metropolitan area. An important aspect in this context is the concept of a Virtual Device (VD): it represents a smart abstracted service built on-the-fly that, leveraging the device resources up to their physical limits, is capable of virtually configuring new devices to satisfy the needs of multiple stakeholders. Let us assume that some high-resolution cameras are monitoring a public square with different perspectives. A VD could add Computer Vision capabilities to the cameras, like object detection and tracking, or a 360 degrees global view, that can be achieved only considering all the cameras as a whole. Similarly, the deployment of many microphones for analyzing noise pollution can be virtualized in one VD able to provide, in real-time, the noise analysis in a public area. A new VD can be created merging the above functionalities to identify crowds or to analyze vehicle traffic. These examples show how, using the same physical infrastructure, it is possible to set up different virtual infrastructures according to the specific applicative needs.

4. Background and Basic Assumptions

In the past years, in order to handle the advent of IoT devices and application constraints in terms of data locality and latency, the Cloud-to-Things Continuum concept has been proposed [26]. The need for efficient services and applications has pushed ICT operators to move a part of their services from the "central" cloud data centers into an intermediate layer, closer to users, defined Edge. In this context, the concept of microservice (MS) or microservice architecture is gaining more and more consensus among both the industrial and academic communities because it allows the development of emerging cutting-edge, efficient Cloud/Edge Computing systems [27]. The MS architecture is a variant of the traditional Service-Oriented Architecture (SOA) that structures an application as a collection of loosely coupled fine-grained services (i.e., microservices) based on lightweight protocols. The benefit of decomposing an application into different smaller services is that it improves modularity, making applications more straightforward and more resilient. Specifically, this architectural style simplifies the deployment of microservices within a flexible and dynamic Cloud/Edge environment that meets their needs. The MS architecture is tailored to the needs of today's software that requires a high degree of flexibility and dynamism, and it allows us to foster the separation among components, creating a more effective environment for building and maintaining highly scalable applications. With the aim of simplifying the management of distributed microservices among several devices, the function as a service model has been proposed.

4.1. Function as a Service

Function as a service (FaaS) is an innovative computational paradigm that provides a platform where users can execute application functionalities without the need to build and maintain the whole infrastructure. In other words, thanks to FaaS, developers can concentrate on the development of small stateless applications able to accomplish specific tasks that will be triggered by predetermined events [28]. In the past years, several cloud-based FaaS solutions, such as AWS Lambda [29], Apache OpenWhisk [30], Azure Functions [31] have been proposed. In parallel to these services, with the aim to "make it simple to turn anything into a serverless function that runs on Linux or Windows through Docker Swarm or Kubernetes", OpenFaas has been proposed [32]. The main advantage of using OpenFaas instead of a cloud-based FaaS solution is the possibility to deploy services on Edge devices distributed across multiple sites [33] and multiple federated networks [34].

4.2. OpenCV

OpenCV (Open Computer Vision) is an open-source computer vision and machine learning software library launched in 1999.

The library consists of more than 2500 optimized algorithms, that vary from face detection to object identification, moving object tracking to image processing for virtual and augmented reality. It is possible to elaborate single pictures or video streams, processing every single captured frame by independently applying filters and image manipulation.

The library currently supports well-established programming languages such as C++, Python, Java and MATLAB interfaces and runs under Windows, Linux, Android and Mac OS.

5. Design

The reference architecture diagram for service provisioning on VD is shown in Figure 1. A general purpose Edge device can be equipped with several sensors or peripheral hardware to provide differentiated services, such as a high-resolution camera, audio capture systems and temperature, humidity or air quality sensors. Several configurable functionalities (implemented according to the FaaS paradigm) can vary from moving object detection to person counting, from vehicle traffic monitoring to license plate recognition, and can be activated on-demand. In particular, services are deployed using containers to take advantage of virtualization technology allowing service portability, resiliency and automatic updates. The FaaS approach is adopted for an easy configuration on-the-fly of the device, especially if switching among different services is required during the time. A configuration file for each service is kept in a local storage area on the device, as well as the raw frames captured from the camera without any elaboration applied. Processed frames and reports are accessible remotely using a Server Message Block (SMB) protocol. This configuration can be replicated for any number of Edge devices available, for example, to cover the city center area. To secure the communication between the Edge device and the endpoint, a Virtual Private Network (VPN) is configured.

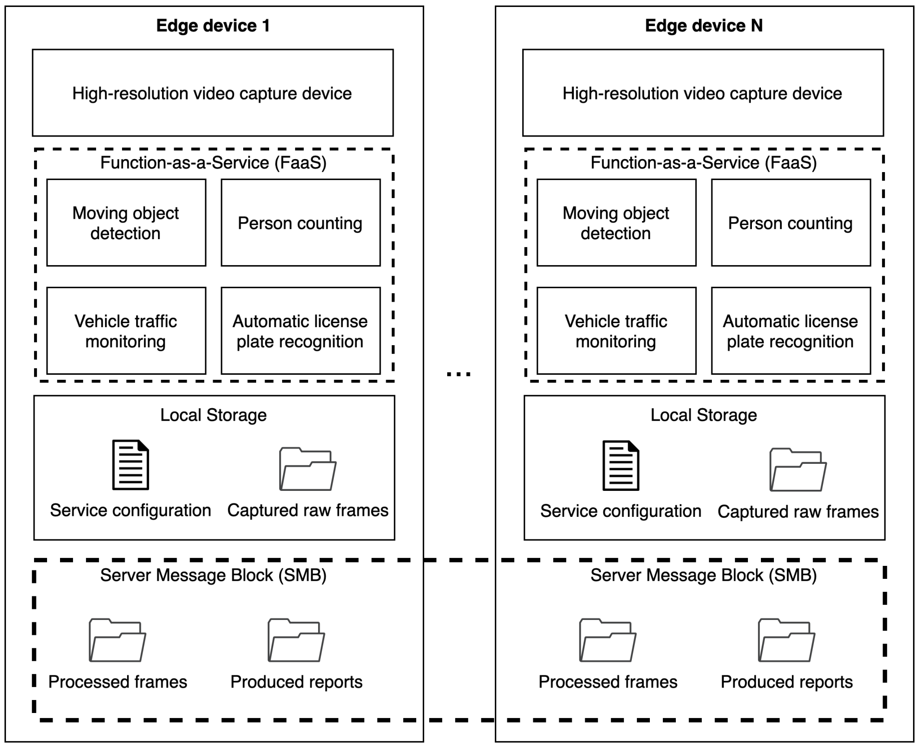

Edge device 1
High-resolution video capture device
Function-as-a-Service (FaaS)
Moving object detection
Person counting
Vehicle traffic monitoring
Automatic license plate recognition
Local Storage
Service configuration
Captured raw frames
Server Message Block (SMB)
Processed frames
Produced reports
...
Edge device N
High-resolution video capture device
Function-as-a-Service (FaaS)
Moving object detection
Person counting
Vehicle traffic monitoring
Automatic license plate recognition
Local Storage
Service configuration
Captured raw frames
Server Message Block (SMB)
Processed frames
Produced reports

Figure 1. Architecture diagram service provisioning on Virtual Device (VD).

In this paper, we present how the proposed solution has been used to support urban mobility management in the city of Messina. In the video processing use case discussed in Section 3.2, we consider a generic video-capturing device connected to a Raspberry Pi 4 which can monitor, record and store single frames or video streams. The last year, Raspberry Pi 4 has been released in the market with impressive hardware characteristics in the MPUs. Nowadays it is possible to buy a fully equipped RPI 4 (4-core CPU, 8GB of RAM, giga-Ethernet, USB3 and USB-C, WIFI and BLE, etc.) for only 70€. This allows us to implement an innovative, scalable and flexible solution at a low cost. The specific workflow we followed for audio and video processing (and that will be further discussed in the next section) was performed through the following steps:

1. Requirement gathering: based on high-level municipality needs, the Edge device is equipped with different hardware systems (camera, microphone, sensors);
2. System configuration: based on specific municipality needs, a configuration file is arranged to indicate the necessary setup information, such as camera resolution, fps, running time and the output report to be produced;
3. System initialisation: this phase depends on the type of service that has to be executed. For instance, considering a traffic monitoring service, a certain number of frames are captured to build the background layer, the minimum size to identify a moving object is adjusted and the exit area is configured (this will be discussed in detail in Section 6); in case of vehicle license plate recognition, the area of interest is cropped to reduce the required computational effort;
4. Service execution: the configured service is run on the Edge device and reports are produced;
5. Report analysis: at any time, the operator can analyze reports produced to plan the required actions for a safer city.

In the next section, implementation details for the described contexts are provided.

6. Implementation

For the implementation of the proposed system, we used the Python3 programming language and the most popular OpenCV (Open Computer Vision) image processing library. This library allows us to apply several filters to the frames acquired by the video source, whether captured via live streaming, video recording or static image, processing each frame as a single independent image.

With reference to the video processing at the Edge scenario, we aimed at controlling vehicles crossing a specific area in the city. To this purpose, a webcam is placed to monitor a specific road junction of the city. An example of a frame captured with a 640 $\times$ 480 resolution is shown in Figure 2. A dedicated Web Server Gateway Interface (WSGI) has been developed with the Python web application framework Flask and can be accessed to look at the captured live stream of the camera device. All the service functionalities have been configured as FaaS to leverage the flexibility of on-demand capabilities based on municipality needs and can be easily switched. All the captured raw frames are saved in a local storage folder and kept for one week (the storage persistence is configurable according to the memory capacity of the Edge device). The processed frames and reports are accessible remotely through the File Transfer Protocol (FTP) or accessing the directory with a network protocol like Samba. The number and effectiveness of the applied manipulation filters can be configured on-the-fly to improve the captured image based on the changing environmental conditions, such as day-night shift or sunny-rainy day. It is important to note that processing time and resource usage at the Edge is not affected by the changing of such filters.

To monitor the number of vehicles moving along a road, we use the Moving Object Detection (MOD) algorithms, and, in particular, MOD algorithms for background subtraction. In particular, during the initialization phase, the system is configured to analyze a specific number of frames. In our experiments, this number is set to 500, but it can be calibrated. This step is required to identify the background composed of atmospheric conditions and natural or artificial ambient light. Frames are acquired and then interpolated to build the so-called background_layer. To track moving objects on the video, it is necessary to subtract the background_layer from the current_frame. The result of the applied filters is a black and white image where only non-static objects are present, as illustrated in Figure 3.

In case of noise caused by the poor image quality and non-static background objects (e.g., car headlights, clouds or tree leaves), these can be removed or reduced by using additional filtering techniques. To obtain a clear frame, the following filters can be applied: opening, closing and dilating. As it can be seen in Figure 3a, tree leaves and cars are identified as objects non-belonging to the background_layer, and these are shown as white pixels in contrast with the static background composed of black pixels. This first level of noise can be reduced by applying the Opening filter (Figure 3b), that is a combination of the Erosion and Dilation filters. It usually removes small noise applying erosion to the resulting frame so the size of the foreground object decreases. On the contrary, the dilate filter used in our experimentation increases the white region in the image so that the foreground object increases to reshape the object size that previously shrank during erosion. The second filtering step is obtained by applying the Closing filter (Figure 3c): it uses the reverse approach with respect to the Opening filter, where the Dilation algorithm is followed by the Erosion one. It is useful for closing small holes inside the foreground objects and helps to further clean the frame thus having a more precise object identification. As the last filtering step, the Dilate filter is applied to join remaining gaps and adjacent objects (Figure 3d).

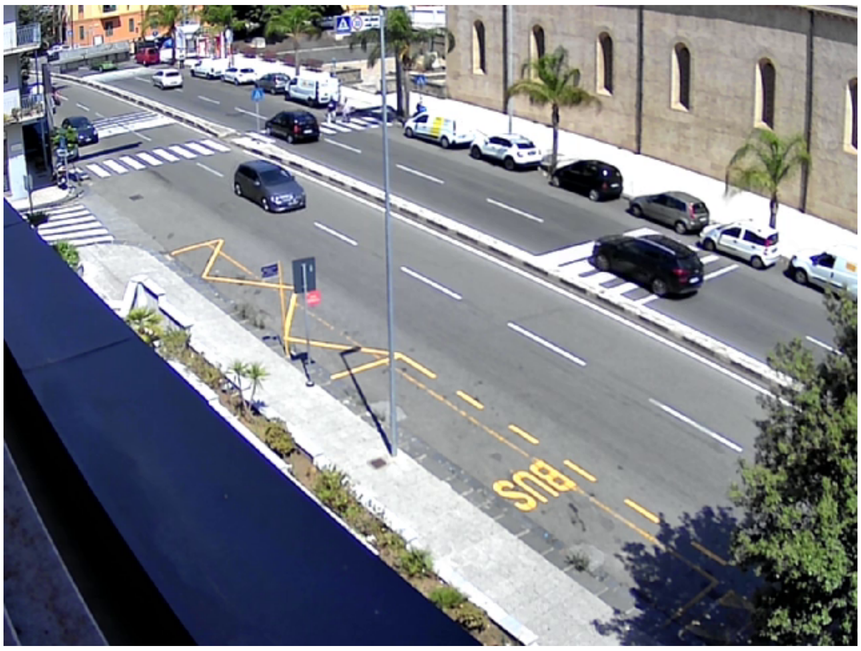


Figure 2. Sample frame captured at 640 × 480 resolution.

To identify the moving objects we are interested in (e.g., cars, trucks, etc.), we need to isolate them from other mobile objects that are not of interest for the urban mobility management use case (e.g., people, cycles, tram wagons, etc.). To this aim, we track the blob in every subsequent frame to validate if the centroid of the minimum surrounding box is moving in the specific area or direction we want to monitor, such as a road junction or intersection. To do so, we setup a virtual area in the captured frame and we monitor if a moving object hits that area. The result of this analysis step is shown in Figure 4, where it is possible to distinguish blue boxes around cars (that are the moving objects we are tracking), the red dot indicating the direction of moving objects (they come from the comparison of consecutive frames) and two green areas that represent the gates where the objects moving in the two opposite directions are counted.

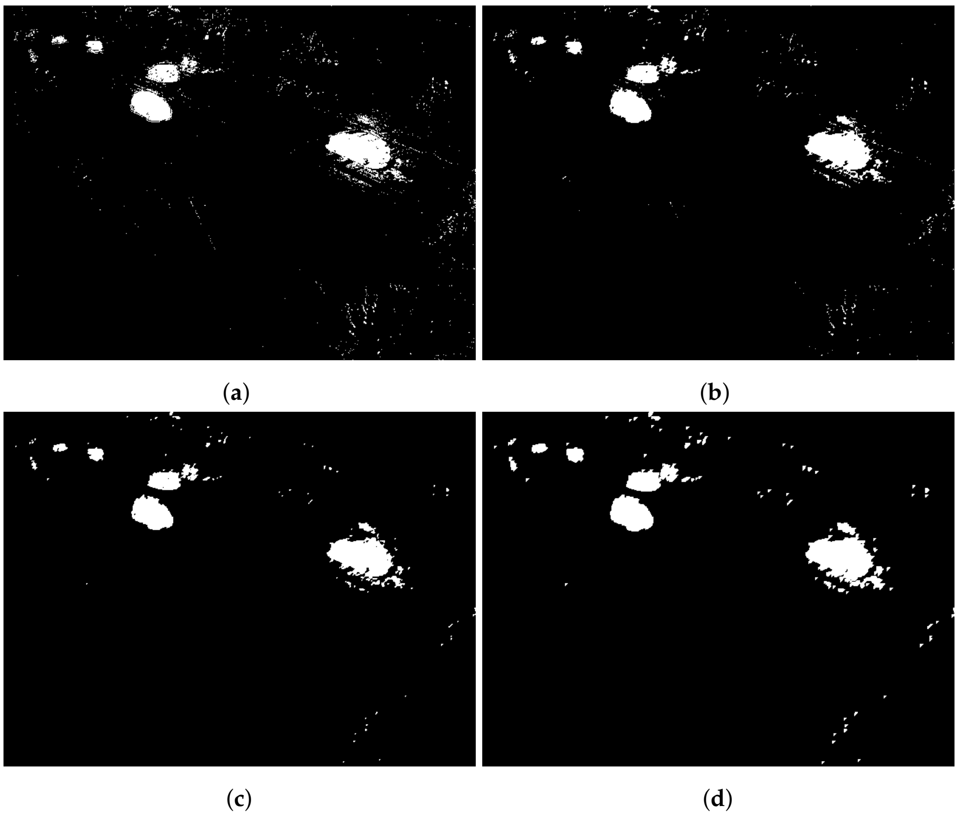


Figure 3. (**a**) Background layer with no filter applied; (**b**) opening filter applied to remove background noise; (**c**) closing filter applied to fill object gaps; (**d**) dilate filter applied to join adjacent objects.

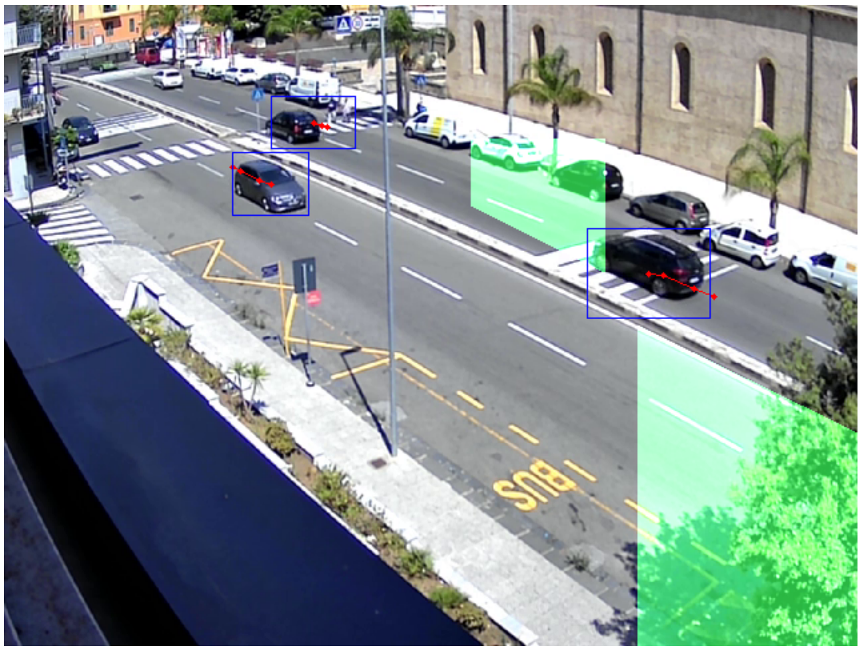


Figure 4. Captured frame with moving objects identified. Objects hitting the green area are counted as moving objects.

The pseudo-code of the described algorithm is provided in the Algorithm 1.

Algorithm 1 Moving object detection algorithm pseudo-code.

```
Class $VehicleCounter$
method $movingObjectDetection(n\_background\_frames, pixel\_coordinates, video\_source)$
  $obj\_coordinates \leftarrow \emptyset$
  $moving\_objects \leftarrow \emptyset$
  $detected\_objects \leftarrow \emptyset$
  $background \leftarrow initBackground(n\_background\_frames)$
  $monitored\_area \leftarrow [pixel\_coordinates]$
  $video\_stream \leftarrow openVideoStream(video\_source)$
  for all $frame \in video\_stream$ do
    $objects \leftarrow (frame - background)$
    $objects \leftarrow applyFilters(objects, [Open, Close, Dilate])$
    for all $object \in objects$ do
      $obj\_coordinates \leftarrow (object, object.position)$
      if $object \in detected\_objects$ then
        for all $position \in obj\_coordinates[object]$ do
          $moving\_object \leftarrow moving\_object \cup position$
        end for
      else
        $detected\_objects \leftarrow detected\_objects \cup object$
      end if
    end for
    if $moving\_object \in monitored\_area$ then
      $vehicle\_count \leftarrow vehicle\_count + 1$
    end if
  end for
```

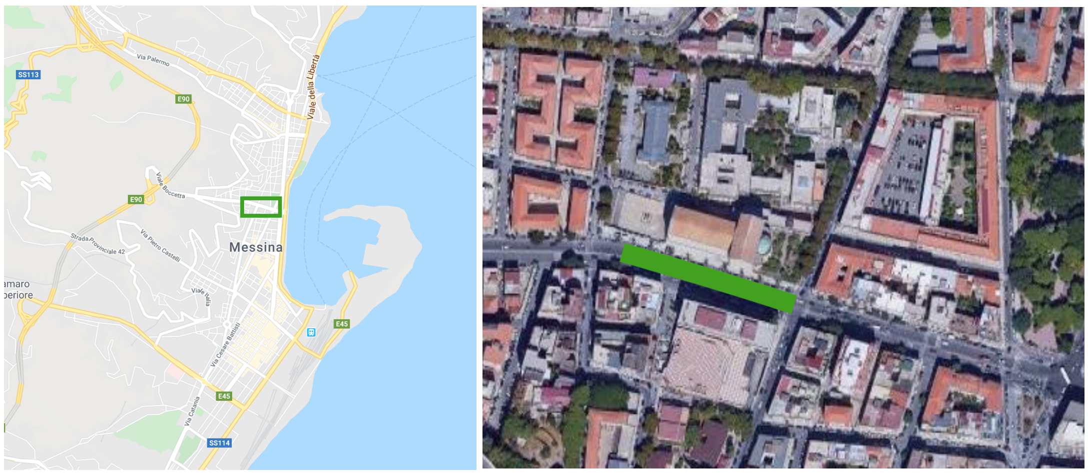
The described analysis can be slightly modified if different requirement are specified, such as if we want to monitor the number of vehicles in a given time slot, or to verify if a vehicle enters a restricted or forbidden transit area or it is moving in the wrong direction. All the above-mentioned scenarios can be easily configured by changing the software function deployed into the Edge device. This is performed exploiting a FaaS approach. For example, it permits to us easily switch the service configuration during different hours of the day (e.g., road access is forbidden during the night and allowed during the day) or depending on special requirements (e.g., road traffic changes during a riot).

7. Performance Evaluation

In this section, we evaluate performance of edge devices to boost cameras from simple surveillance device to general-purposes device able to monitor the road traffic. In particular, we make two different types of analyses to test both the feasibility and the applicability of these devices for such purposes. More specifically, we make both a quantitative analysis comparing the performance, in terms of vehicle counting, of a Traficam with a Raspberry Pi 4 and qualitative analysis in order to test the flexibility of edge devices for supporting different tasks and workloads.

7.1. Reference Scenario

As we discussed in previous sections, within the Mesmart project, the Municipality of Messina is monitoring the traffic of main roads. In our paper, we consider as reference scenario Viale Boccetta, one of the main roads that link the highway to the touristic harbor. Figure 5a displays the map of Messina, while Figure 5b shows a zoom in the area of Viale Boccetta. In particular, Figure 5b shows the monitored zone presented in this paper in green color.



(**a**) (**b**)

Figure 5. (**a**) Plan of the City of Messina (**left**); (**b**) detail of the monitored zone (**right**).

In this scenario, in order to test the feasibility and the applicability of edge devices for vehicle counting, we compared performances of a Traficam with a Raspberry Pi 4, equipped with a Full HD webcam, installed at "PalaAntonello". Hardware and software characteristics of these devices are summarised in Table 1:

Table 1. Testbed characteristics.

Parameter	**Traficam**	**Raspberry**
CMOS type	1/4" color	1/3" color 2 MP
Resolution	640 × 480 pixels (VGA)	1920 × 1080 pixels (Full HD)
Frame Rate	25 fps	30 fps
Detection Zones	24	unlimited
CPU	N/A	Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5 GHz
RAM	N/A	4 GB
OS	GNU	Raspbian

7.2. Investigation on Video Functionalities

In this section, we investigate the service quality of a general-purpose platform in comparison with a system built and dedicated to the specific task of vehicle counting. In particular, considering a one-week monitoring, we divided the business days into six different time slots:

1. night: 01:30–5:30;
2. sunrise: 5:30–9:30;
3. morning: 09:30–13:30;
4. afternoon: 13:30–17:30;
5. sunset: 17:30–21:30;
6. evening: 21:30–01:30.

These time slots have been chosen to take into account different light conditions on the road. Per each time slot, we calculated the average number of vehicles that transit in both directions (West–East and East–West). In order to validate the outcome of the deployed system, per each time slot and direction, we observed several samples of the recorded video in order to calculate the number of false-positive, false-negative and the accuracy of both the systems.

In Figure 6, the behavior during the time slot "night" is shown. As the reader can observe, the performance of both devices is comparable. However, observing the video, we noticed that the accuracy of both devices is quite low. This is because the resolution of the Traficam is quite low and, therefore, some vehicles, especially the dark ones, are not recognized. With reference to the Raspberry Pi device, our algorithm is not able to distinguish vehicles from their own headlights and, thus, it does not increment the vehicle counter. With the increasing of illumination, the behavior is different. Starting from 04:30 a.m., the number of vehicles detected from the raspberry, especially in the direction W-E, increases.

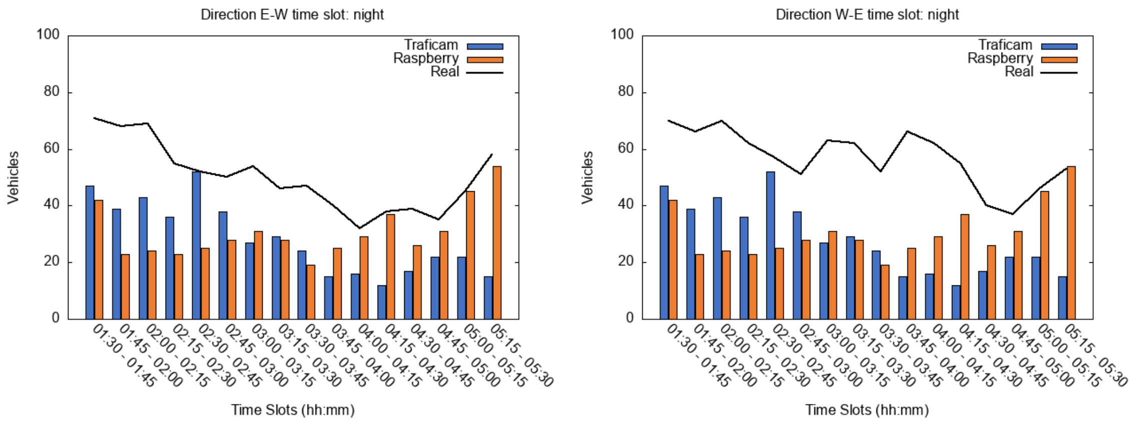


Figure 6. Comparison of Traficam (in blue) and Edge device (in orange) for vehicle counting during the night time slot. On the left part of the figure the direction E-W is shown, on the right part the direction W-E is displayed.

This is much more evident during the time slot "sunrise" shown in Figure 7. In fact, issues due to headlights disappear and the raspberry is able to detect vehicles with an accuracy of 90%. During this time slot, light conditions are still not sufficient for Traficam. In fact, the number of vehicles detected by the Traficam becomes comparable to the Raspberry's one only at the end of this time slot, starting from about 9:00 a.m.

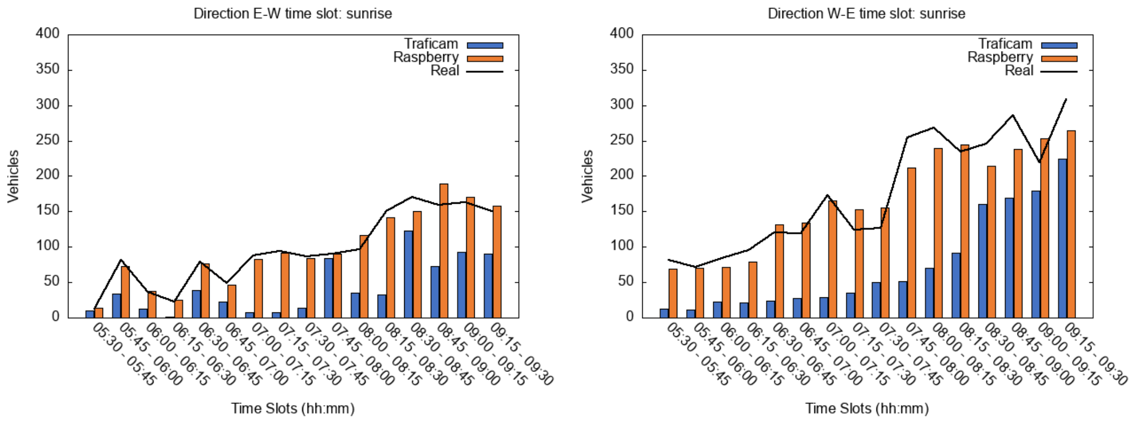


Figure 7. Comparison of Traficam (in blue) and Edge device (in orange) for vehicle counting during the sunrise time slot. On the left part of the figure, the direction E-W is shown, on the right part the direction is W-E.

In the time slot "morning", as the reader can observe in Figure 8, we can distinguish different behaviors. In particular, chart "E-W" shows two different behaviors depending on the position of the sun. From 9:30 to 10:15 the Raspberry-based system is able to detect more vehicles with respect to the Traficam. In contrast, since 10:15 the light conditions are good for both devices. There are only a few discrepancies in the number of vehicles detected by Traficam and Raspberry respectively. This is due to the presence of big vehicles such as trucks or buses and cars crossing two lanes. Considering the direction "W-E" is it possible to observe three different behaviors: (i) from 9:30 to 10:00 the

Raspberry-based system is able to detect more vehicles with respect to the Traficam; (ii) considering the time period 10:00–11:30 the light conditions are good for both devices and we can note only a few discrepancies; (iii) instead, starting from 11:30, the light conditions are not ideal for the Traficam, in fact, as we can observe in the figure, there are considerable differences between the results of both devices. As we will discuss for the "afternoon" time slot, these discrepancies depend on the light conditions.

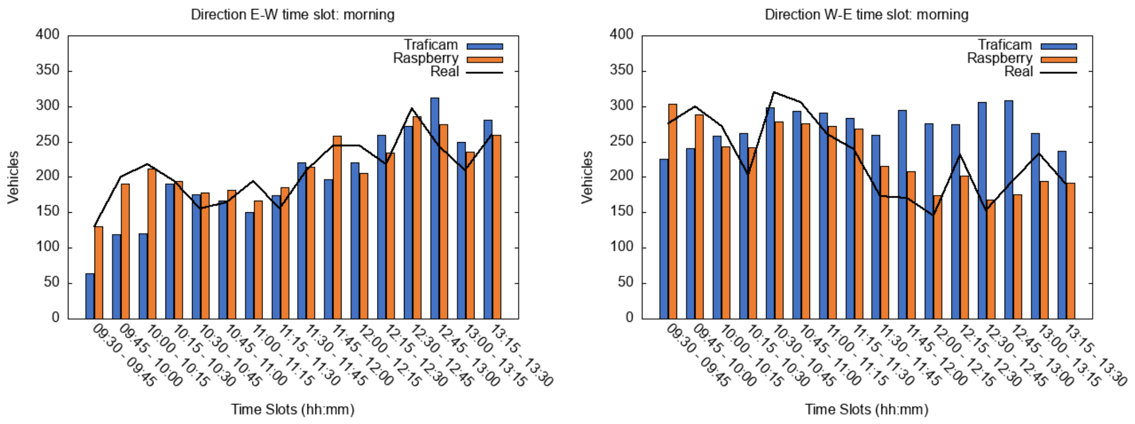


Figure 8. Comparison of Traficam (in blue) and Edge device (in orange) for vehicle counting during the morning time slot. On left part of the figure the direction E-W is shown, on the right part the direction is W-E.

During the time slot "afternoon", as the reader can observe in Figure 9, there is a big difference in vehicles counting especially for E-W direction at 2:00 p.m. Observing the video we noticed that, due to light conditions, one of the Traficam's detectors was blocked at the active position thus the vehicle counter was incremented constantly.

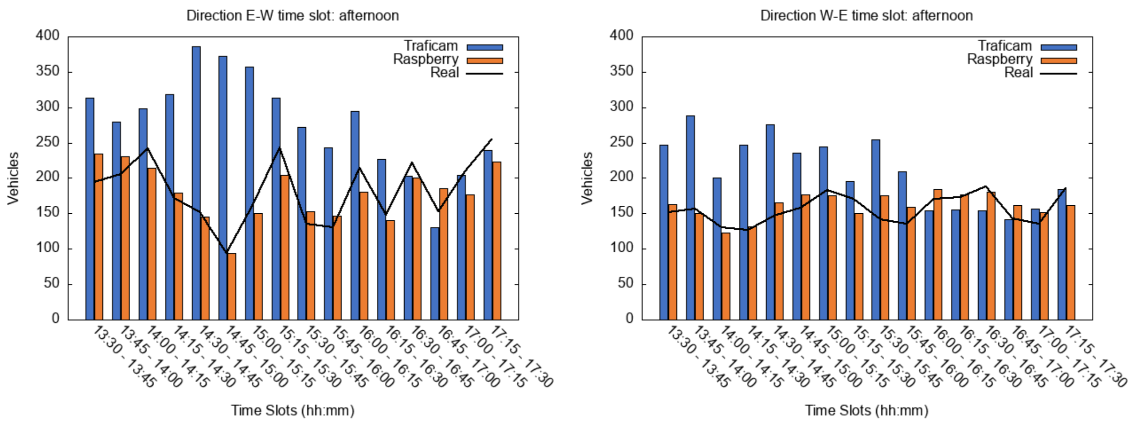


Figure 9. Comparison of Traficam (in blue) and Edge device (in orange) for vehicle-counting during the afternoon time slot. On left part of the figure the direction E-W is shown, on the right part the direction is W-E.

In Figure 10 the behavior of both devices during the time slot "sunset" is shown. The accuracy of both devices is quite good for a great part of the time slot. The only big difference is in the direction E-W after 8:00 p.m. As the reader can observe in the figure, due to the absence of environmental light, the vehicle headlights become more prominent and the Raspberry is not able to distinguish the vehicles.

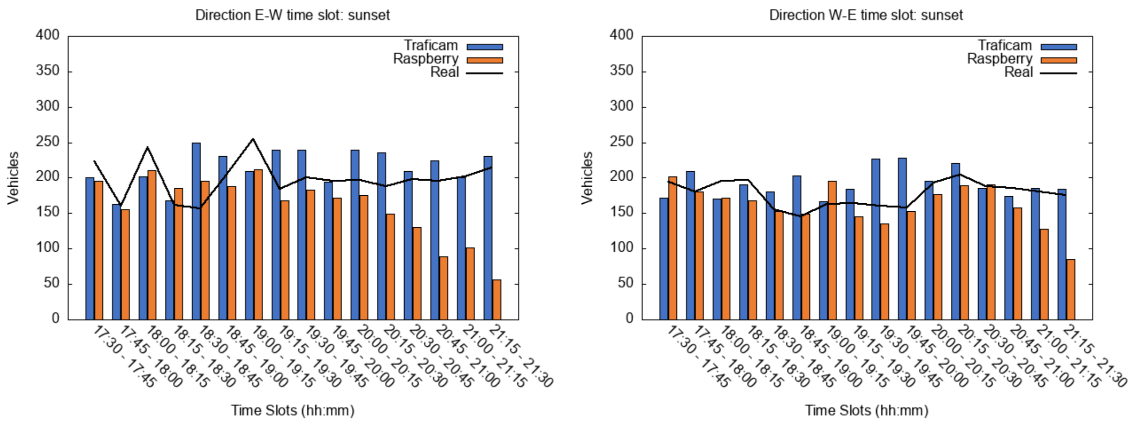


Figure 10. Comparison of Traficam (in blue) and Edge device (in orange) for vehicle counting during the sunset time slot. On left part of the figure, the direction E-W is shown, on the right part the direction is W-E.

Finally, in Figure 11 the behavior of both devices during the time slot "evening" is shown. During this time slot, we can observe two different behaviors: in the first part of the chart (till 10:30 p.m.) the behavior is very similar to the "sunset" time slot, in fact, the Traficam detects vehicles that are not visible for the Raspberry. Instead, in the second part of the chart, the behavior of both devices is similar to the "night" time slot. Both devices detect almost the same number of vehicles but, in both cases, observing the video we can see that a great part of vehicles remains undetected.

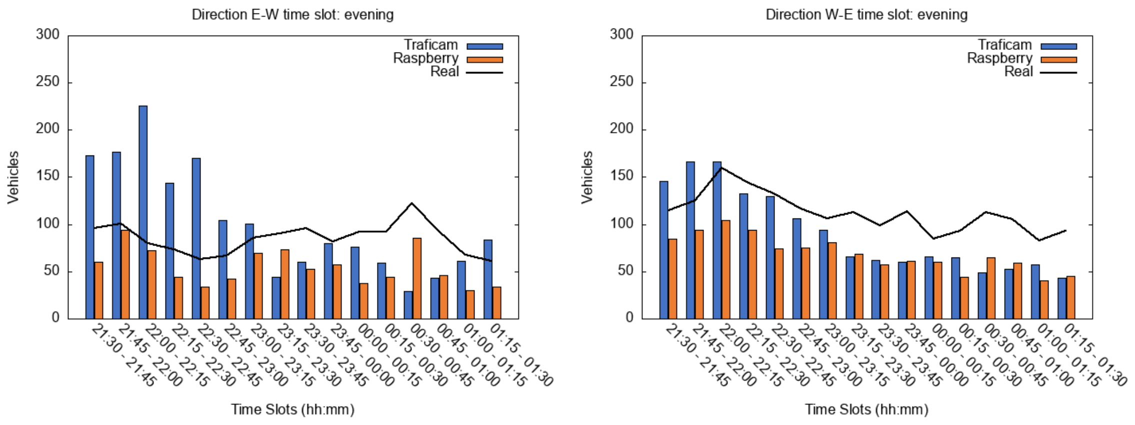


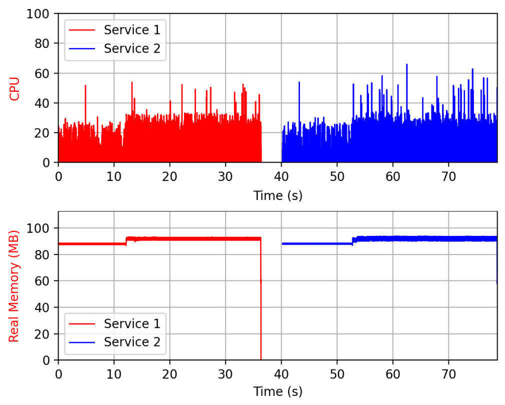
Figure 11. Comparison of Traficam (in blue) and Edge device (in orange) for vehicle counting during the evening time slot. On left part of the figure the direction E-W is shown, on the right part the direction is W-E.

Analyzing the outcomes of both devices, we found that the system based on Raspberry does not over-count the number of vehicles that pass through the monitored area. In particular, it performs better during the daytime reaching 95% of accuracy. During the nighttime, the system is not able to distinguish between vehicles and headlights therefore the accuracy of the system decreases. Different behaviors can be seen for the Traficam. In fact, due to the presence of multiple static detectors, it over-counts the number of vehicles especially around 2:00 p.m.

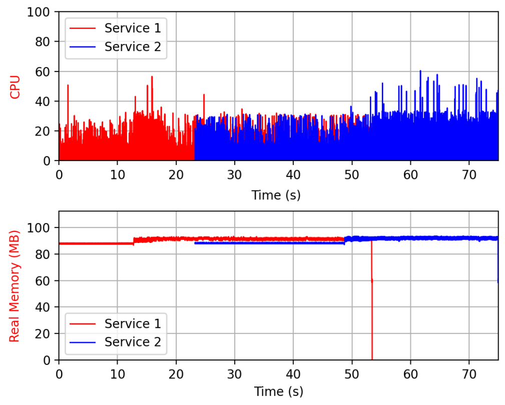
7.3. Performance Assessment

Performance assessment is aimed at understanding possible concrete usage at the Edge (e.g., RPI 4), a platform conceived for hosting any kind of data processing (e.g., Audio, Video, Sensing, etc.) is able to fulfill several computation ad storage requirements. In Section 7.2 we investigated the service quality that a general-purpose platform can have in comparison with a system born and dedicated to a specific task. Here, we want to understand if a general-purpose platform can have enough hardware resources to come up with more tasks at the same time. We tested the device to run two different services of vehicle counting simulating the on-demand request that could be made by the Municipality stakeholders. In the first scenario, we simulate the case where there is a need to switch from one service to another. As indicated in Figure 12a the down-time is minimal and in just a few seconds the Service 2 is operative. The second scenario in Figure 12b presents a similar case but the two services co-exist for a certain period before Service 1 is switched off. As demonstrated, the device is able to manage both services simultaneously.

In addition to the vehicle counting task, we stressed the Raspberry performing the OCR on frames for the number plate recognition considering 1, 2, 4 and 10 cameras simultaneously connected to the same devices. As the reader can observe in Figure 13, the execution time increases almost linearly within the number of cameras. Considering our urban scenario, we can consider accepting the time required for processing up to four cameras on the same device.



(**a**)



(**b**)

Figure 12. Resource comparison for two different services running on the general-purpose device. (**a**) two services running with a minimum downtime; (**b**) two services running simultaneously.

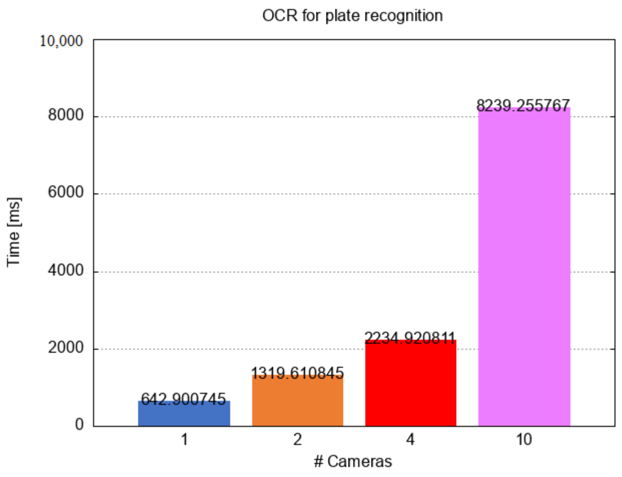


Figure 13. Execution time for performing the OCR (Optical Character Recognition) for number plate recognition on Raspberry considering 1, 2, 4 and 10 cameras connected to the same device.

8. Conclusions and Future Works

In this paper, starting from the requirements of the Italian project Mesmart, we investigated the use of a single edge device for accomplishing multiple tasks in the domain of video processing instead of buying a specific camera per each task. In particular, we implemented a FaaS solution that allows the users to run several tasks on the same device such as vehicle counting, number plate recognition, object identification etc. Our solution is composed of a full-HD webcam connected to the Raspberry Pi 4 running OpenCV algorithms. In order to validate our system, we made two different types of analysis: a qualitative analysis comparing the performance of the Raspberry with a Traficam, a device specifically designed for vehicle counting, and a quantitative analysis running several concurrent tasks on Raspberry in order to discover if it has enough resources for managing multiple tasks. Results demonstrated that the Raspberry-based system, during the daytime, performs better than Traficam, in particular, it never over-counts vehicles, reaching an accuracy of around 95%. Different behavior can be seen during the nighttime. In this case, both devices present almost the same performance in terms of vehicle counting, but the accuracy is quite low. Finally, in order to test the applicability of edge devices for multiple purposes, in addition to vehicle counting, we ran the OCR for number plate recognition considering 1, 2, 4 and 10 concurrent cameras. From this analysis, we discovered that considering an urban scenario the Raspberry is able to manage up to four cameras. In future works, we plan to improve the vehicle counting technique in order to increase the accuracy of the edge-based approach during the night time.

Author Contributions: Conceptualization, M.V., G.D., M.F. and A.G.; methodology, M.V., A.G. and A.R.; software, A.G. and A.R.; validation, A.G. and A.R.; writing—original draft preparation, A.G., A.R., M.F. and G.D.; writing—review and editing, M.F., M.V. and G.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Italian National Operational Program (PON) Metropolitan City 2014-2020—"MeSmart", CUP F41I18000230006.

Acknowledgments: The authors thank the Master's students Alessio Catalfamo, Valeria Lukaj, Francesco Martella and Christian Sicari for their valuable work on the vehicle counting process.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Filocamo, B.; Galletta, A.; Fazio, M.; Ruiz, J.; Sotelo, M.; Villari, M. *An Innovative Osmotic Computing Framework for Self Adapting City Traffic in Autonomous Vehicle Environment*; IEEE: Piscataway, NJ, USA, 2018; pp. 1267–1270.
2. ISTAT. Maritime Transport: Passengers for Each Port by Direction. 2020. Available online: http://dati.istat.it/index.aspx?queryid=25765 (accessed on 26 November 2020).
3. Il porto di Messina in vetta alle classifiche nazionali per traffico. Available online: https://normanno.com/attualita/porto-messina-classifiche-nazionali-traffico-passeggeri/ (accessed on 26 November 2020).
4. Nagmode, V.S.; Rajbhoj, S.M. An IoT Platform for Vehicle Traffic Monitoring System and Controlling System Based on Priority. In Proceedings of the 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA), Pune, India, 17–18 August 2017; pp. 1–5.
5. Darbari, M.; Yagyasen, D.; Tiwari, A. Intelligent Traffic Monitoring Using Internet of Things (IoT) with Semantic Web. In *Emerging ICT for Bridging the Future, Proceedings of the 49th Annual Convention of the Computer Society of India (CSI) Volume 1*; Satapathy, S.C., Govardhan, A., Raju, K.S., Mandal, J.K., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 455–462.
6. Parekh, S.; Dhami, N.; Patel, S.; Undavia, J. Traffic Signal Automation Through IoT by Sensing and Detecting Traffic Intensity Through IR Sensors. In *Information and Communication Technology for Intelligent Systems*; Satapathy, S.C., Joshi, A., Eds.; Springer: Singapore, 2019; pp. 53–65.
7. Barthélemy, J.; Verstaevel, N.; Forehead, H.; Perez, P. Edge-Computing Video Analytics for Real-Time Traffic Monitoring in a Smart City. *Sensors* **2019**, *19*, 2048. [CrossRef] [PubMed]
8. Mehrabi, A.; Siekkinen, M.; Ylä-Jaaski, A. QoE-Traffic Optimization Through Collaborative Edge Caching in Adaptive Mobile Video Streaming. *IEEE Access* **2018**, *6*, 52261–52276. [CrossRef]
9. Nikouei, S.Y.; Chen, Y.; Song, S.; Faughnan, T.R. Kerman: A Hybrid Lightweight Tracking Algorithm to Enable Smart Surveillance as an Edge Service. In Proceedings of the 2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC), Vegas, NV, USA, 11–14 January 2019; pp. 1–6.

10. Zhao, Y.; Yin, Y.; Gui, G. Lightweight Deep Learning based Intelligent Edge Surveillance Techniques. *IEEE Trans. Cogn. Commun. Netw.* **2020**. [CrossRef]
11. Fazio, M.; Paone, M.; Puliafito, A.; Villari, M. Huge amount of heterogeneous sensed data needs the cloud. In Proceedings of the International Multi-Conference on Systems, Signals & Devices, Chemnitz, Germany, 20–23 March 2012; pp. 1–6.
12. Khan, L.U.; Yaqoob, I.; Tran, N.H.; Kazmi, S.M.A.; Dang, T.N.; Hong, C.S. Edge Computing Enabled Smart Cities: A Comprehensive Survey. *IEEE Internet Things J.* **2020**, *7*, 10200–10232. [CrossRef]
13. Xu, X.; Huang, Q.; Yin, X.; Abbasi, M.; Khosravi, M.R.; Qi, L. Intelligent Offloading for Collaborative Smart City Services in Edge Computing. *IEEE Internet Things J.* **2020**, *7*, 7919–7927. [CrossRef]
14. Kafıloğlu, S.S.; Gür, G.; Alagöz, F. Cooperative Caching and Video Characteristics in D2D Edge Networks. *IEEE Commun. Lett.* **2020**, *24*, 2647–2651. [CrossRef]
15. Wang, D.; Peng, Y.; Ma, X.; Ding, W.; Jiang, H.; Chen, F.; Liu, J. Adaptive Wireless Video Streaming Based on Edge Computing: Opportunities and Approaches. *IEEE Trans. Serv. Comput.* **2019**, *12*, 685–697. [CrossRef]
16. Zhao, Y.; Chen, Q.; Cao, W.; Jiang, W.; Gui, G. Deep Learning Based Couple-like Cooperative Computing Method for IoT-based Intelligent Surveillance Systems. In Proceedings of the IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Istanbul, Turkey, 8–11 September 2019; pp. 1–4.
17. Liu, M.; Yu, F.R.; Teng, Y.; Leung, V.C.M.; Song, M. Distributed Resource Allocation in Blockchain-Based Video Streaming Systems With Mobile Edge Computing. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 695–708. [CrossRef]
18. Okaishi, W.A.; Atouf, I.; Benrabh, M. Real-Time Traffic Light Control System Based on Background Updating and Edge Detection. In Proceedings of the 2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS), Fez, Morocco, 3–4 April 2019; pp. 1–5.
19. Yang, K.; Zhou, Y.; Han, S.; Fang, Y.; Ma, X.; Zhou, J.; Yao, K. Video-Based Traffic Flow Monitoring Algorithm for Single Phase Position at An Intersection. In Proceedings of the 2019 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC), Guangzhou, China, 20–23 December 2019; pp. 101–105.
20. Wang, Z.; Cheng, G.; Zheng, J. Road Edge Detection in All Weather and Illumination via Driving Video Mining. *IEEE Trans. Intell. Veh.* **2019**, *4*, 232–243. [CrossRef]
21. Baresi, L.; Filgueira Mendonça, D. Towards a Serverless Platform for Edge Computing. In Proceedings of the 2019 IEEE International Conference on Fog Computing (ICFC), Prague, Czech Republic, 24–26 June 2019; pp. 1–10.
22. Gadepalli, P.K.; Peach, G.; Cherkasova, L.; Aitken, R.; Parmer, G. Challenges and Opportunities for Efficient Serverless Computing at the Edge. In Proceedings of the 2019 38th Symposium on Reliable Distributed Systems (SRDS), Lyon, France, 1–4 October 2019; pp. 261–2615.
23. Hussain, R.F.; Salehi, M.A.; Semiari, O. Serverless Edge Computing for Green Oil and Gas Industry. In Proceedings of the 2019 IEEE Green Technologies Conference(GreenTech), Lafayette, LA, USA, 3–6 April 2019; pp. 1–4.
24. Palade, A.; Kazmi, A.; Clarke, S. An Evaluation of Open Source Serverless Computing Frameworks Support at the Edge. In Proceedings of the 2019 IEEE World Congress on Services (SERVICES), Milan, Italy, 8–13 July 2019; Volume 2642-939X, pp. 206–211.
25. Mulfari, D.; Celesti, A.; Fazio, M.; Villari, M.; Puliafito, A. *Using Google Cloud Vision in Assistive Technology Scenarios*; IEEE: Piscataway, NJ, USA, 2016; pp. 214–219.
26. De Donno, M.; Tange, K.; Dragoni, N. Foundations and Evolution of Modern Computing Paradigms: Cloud, IoT, Edge, and Fog. *IEEE Access* **2019**, *7*, 150936–150948. [CrossRef]
27. Liu, F.; Tang, G.; Li, Y.; Cai, Z.; Zhang, X.; Zhou, T. A Survey on Edge Computing Systems and Tools. *Proc. IEEE* **2019**, *107*, 1537–1562. [CrossRef]
28. Sewak, M.; Singh, S. *Winning in the Era of Serverless Computing and Function as a Service*; IEEE: Piscataway, NJ, USA, 2018; pp. 1–5.
29. Amazon AWS Lambda. Available online: https://aws.amazon.com/it/lambda/ (accessed on 25 August 2020).
30. Apache Openwhisk. Available online: https://openwhisk.apache.org/ (accessed on 25 August 2020).
31. Microsoft Azure Functions. Available online: https://azure.microsoft.com/en-us/services/functions/ (accessed on 25 August 2020).

32. What Is OpenFaaS and How Can It Drive Innovation? An Interview with Creator Alex Ellis. Available online: https://www.contino.io/insights/what-is-openfaas-and-why-is-it-an-alternative-to-aws-lambda-an-interview-with-creator-alex-ellis (accessed on 25 August 2020).
33. Buzachis, A.; Fazio, M.; Celesti, A.; Villari, M. *Osmotic Flow Deployment Leveraging FaaS Capabilities*; Springer: Berlin, Germany, 2019; pp. 391–401.
34. Moreno-Vozmediano, R.; Huedo, E.; Llorente, I.; Montero, R.; Massonet, P.; Villari, M.; Merlino, G.; Celesti, A.; Levin, A.; Schour, L.; et al. BEACON: A cloud network federation framework. *Commun. Comput. Inf. Sci.* **2016**, *567*, 325–337. [CrossRef]