UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Object detection and counting with computer vision: A comparative analysis of cloud and edge solutions in terms of cost, quality, and performance

Nilson Tinassi Peres

Dissertação de Mestrado do Programa de Mestrado Profissional em Matemática, Estatística e Computação Aplicadas à Indústria (MECAI)

ICMC USP SÃO CARLOS

Nilson Tinassi Peres

Object detection and counting with computer vision: A comparative analysis of cloud and edge solutions in terms of cost, quality, and performance

Dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP – in accordance with the requirements of the Professional Master's Program in Mathematics Statistics and Computing Applied to Industry, for the degree of Master in Science. *EXAMINATION BOARD PRESENTATION COPY*

Concentration Area: Mathematics, Statistics and Computing

Advisor: Profa. Dra. Gleici da Silva Castro Perdona

USP – São Carlos
July 2025

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

PERES, NILSON TINASSI
P434d Detecção e contagem de objetos com visão computacional: Uma análise comparativa entre soluções em nuvem e em borda nos aspectos de custo, qualidade e desempenho / NILSON TINASSI PERES; orientadora GLEICI DA SILVA CASTRO PERDONA. -- São Carlos, 2025.
146 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Mestrado Profissional em Matemática, Estatística e Computação Aplicadas à Indústria) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2025.

1. Computação em Nuvem. 2. Processamento em Borda. 3. Visão Computacional. 4. Detecção e Contagem de Objetos. I. CASTRO PERDONA, GLEICI DA SILVA, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Nilson Tinassi Peres

Detecção e contagem de objetos com visão computacional: Uma análise comparativa entre soluções em nuvem e em borda nos aspectos de custo, qualidade e desempenho

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre – Mestrado Profissional em Matemática, Estatística e Computação Aplicadas à Indústria. *EXEMPLAR DE DEFESA*

Área de Concentração: Matemática, Estatística e Computação

Orientadora: Profa. Dra. Gleici da Silva Castro Perdona

USP – São Carlos
Julho de 2025

Dedico a minha amada Luna.

ACKNOWLEDGEMENTS

Agradeço a Jeová Deus, o dador da vida e que merece toda glória (Salmos 96:8), por ter me habilitado a escrever esse trabalho e a conciliar as muitas responsabilidades que fazem parte da minha vida. Sem essa ajuda especial, acredito que não teria conseguido. Realmente, para todas as coisas tenho forças, graças àquele que me dá poder (Filipenses 4: 13).

Também, agradeço à minha fiel companheira, Luana, que sempre me apoiou nos meus sonhos e sempre me ajudou a carregar as minhas cargas. A vida é feita de alegrias e tristezas, altos e baixos, mas você esteve ao meu lado em todos os momentos, fossem bons, fossem maus. Pode ser fácil vencer a um que está só, mas dois juntos podem resistir... (Eclesiastes 4: 12). Sua companhia, seu amor, me dão forças para resistir. Eu te amo meu amor e espero o dia em que vamos viver para sempre, porque feliz, eu já sou, e ao seu lado, sempre vou ser.

Finalmente, agradeço aos meus pais, Rosana e Nilson, que têm me amado desde o primeiro dia. Pai, mãe, se vocês sentem qualquer orgulho de mim hoje, vocês podem sentir orgulho de si mesmos, pois se eu me tornei, quem eu sou hoje, é graças ao que vocês me ensinaram (Provérbios 22: 6). Eu amo vocês.

"... não desista de fazer o que é bom,
pois colherá no tempo devido,
se não desanimar."
(Gálatas 6: 9, Tradução do Novo Mundo da Bíblia Sagrada)

RESUMO

PERES, N. T. **Detecção e contagem de objetos com visão computacional: Uma análise comparativa entre soluções em nuvem e em borda nos aspectos de custo, qualidade e desempenho**. 2025. 134 p. Dissertação (Mestrado – Mestrado Profissional em Matemática, Estatística e Computação Aplicadas à Indústria) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2025.

A evolução das técnicas de visão computacional têm permitido desenvolvimento de aplicações inovadoras. Também, provedores de computação em nuvem como Amazon, Microsoft e Google, têm disponibilizado soluções avançadas em suas plataformas. Mas, essas plataformas não tornaram as soluções embarcadas obsoletas, já que assim essas soluções também evoluíram no desempenho, mas evoluíram principalmente para se tornarem mais acessíveis quanto ao conhecimento necessário para usá-las.

Dado o diferente número de provedores, serviços disponíveis e modelos de inteligência artificial, levanta um dilema: qual solução é mais adequada para resolver um problema real? Não há uma resposta definitiva. Dois problemas similares, que se diferenciam talvez por um único requerimento, podem requerer abordagens essencialmente diferentes. Por isso, esse estudo introduz um problema real e analisa como diferentes requisitos podem influenciar em decisões estratégicas sobre quais arquiteturas (em nuvem ou em borda), plataformas (AWS, Azure ou GCP), modelos e tecnologias utilizar.

O problema exemplo em questão se trata da detecção e contagem de pessoas e veículos - um tipo de problema exemplo recorrentemente estudado - viabilizando a comparação dos resultados aqui apresentados e reduzindo o viés de análise do autor. Também, esse problema exemplo é tangível, permitindo melhor compreensão de objetivos, requisitos, metodologias e conclusões. A avaliação considera 3 parâmetros: custo, qualidade e desempenho.

Os parâmetros escolhidos para avaliação permitem uma visão holística das opções, extrapolando o problema exemplo, e servindo como base para tomada de decisão assertiva em um mercado competitivo. Tais parâmetros são avaliados nesse estudo por meio de diferentes experimentos como análise de latência, análise de robustez à condições variáveis, análise de custo, análise da qualidade e análise de processamento em tempo-real.

Espera-se determinar qual a solução mais apropriada para o problema exemplo, porém simultaneamente demonstrar quais as forças e fraquezas de outras soluções disponíveis. Desse modo, este estudo contribui ao fornecer diretrizes para escolha de soluções (em nuvem ou em borda) para aplicações de visão computacional.

Palavras-chave: Soluções em nuvem, Processamento em borda, Visão computacional, Comparação, Contagem de objetos.

ABSTRACT

PERES, N. T. **Object detection and counting with computer vision: A comparative analysis of cloud and edge solutions in terms of cost, quality, and performance**. 2025. 134 p. Dissertação (Mestrado – Mestrado Profissional em Matemática, Estatística e Computação Aplicadas à Indústria) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2025.

The evolution of computer vision techniques has enabled the development of innovative applications. At the same time, major cloud providers — Amazon, Microsoft, and Google — have integrated advanced computer vision solutions into their platforms. However, this progress has not rendered embedded solutions obsolete. On the contrary, such solutions have significantly improved in performance and, more importantly, have become more accessible in terms of the expertise required to operate them.

Given the wide range of providers, available services, and AI models, a recurring dilemma emerges: which solution is best suited to solve a real-world problem? There is no definitive answer. Even two seemingly similar problems may require fundamentally different approaches due to a single distinct requirement. Therefore, this study introduces a real-world problem and analyzes how specific requirements can influence strategic decisions regarding architectures (edge or cloud-based), platforms (AWS, Azure, or GCP), models, and technologies.

The selected case study — object detection and counting of people and vehicles — is a well-established problem in the field, allowing for comparison with existing solutions and minimizing the author's analytical bias. Its tangible nature also facilitates a clearer understanding of objectives, requirements, methodologies, and conclusions. The evaluation considers three key parameters: cost, quality and performance.

These parameters provide a holistic view of the available options, going beyond the case study itself and serving as a foundation for informed decision-making in competitive markets. They are assessed through a series of experiments involving latency analysis, robustness under variable conditions, cost analysis, quality evaluation, and real-time processing capability.

The goal is to determine the most suitable solution for the case study while also highlighting the strengths and weaknesses of alternative approaches. This study ultimately contributes by offering practical guidelines for selecting between cloud-based and edge architectures in computer vision applications.

Keywords: Cloud solutions, Edge computing, Computer vision, Comparative analysis, Object counting.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

ACF	*Aggregated Channel Features*
API	*Application Programming Interface*
AWS	*Amazon Web Services*
CI/CD	*Continuous Integration/Continuous Deployment*
COCO	*Common Object in Context*
CSP	*Cloud Service Provider*
CSPNet	*Cross Stage Partial Network*
DeepSORT	*Deep Simple Online and Realtime Tracking*
ECR	*Elastic Container Registry*
FLOPs	*Floating Point Operations*
FREAK	*Fast Retina Keypoint*
GCP	*Google Cloud Platform*
HOG	*Histogram of Oriented Gradients*
HTTP	*Hypertext Transfer Protocol*
IaaS	*Infrastructure as a Service*
LIDAR	*Light Detection and Range*
mAPval 50-95	*mean Average Precision*
MOT	*Multiple Object Tracking*
MOTA	*Multiple Object Tracking Accuracy*
MOTP	*Multiple Object Tracking Precision*
NA	*not applicable*
NLP	*Natural Language Processing*
OCR	*Optical Character Recognition*
ORB	*Oriented FAST and Rotated BRIEF*
PaaS	*Platform as a Service*
PANet	*Path Aggregation Network*
R-CNN	*Region-based Convolutional Neural Network*
SaaS	*Software as a Service*
SIFT	*Scale Invariant Feature Transform*
SLA	*Service Level Agreement*
SORT	*Simple Online and Realtime Tracking*

SQL	*Structured Query Language*
SSD	*Single Shot MultiBox Detector*
SVM	*Support Vector Machine*
ViT	*Vision Transformer*
YOLO	*You Only Look Once*

CONTENTS

CHAPTER

1

INTRODUÇÃO

The advancement of computer vision techniques has driven the development of intelligent applications in areas such as security, urban mobility, retail, logistics, and agriculture. This progress has been enabled by the evolution of algorithms and the popularization of accessible computational resources.

In parallel, the main cloud computing providers — *Amazon Web Services* (AWS), Microsoft Azure, and *Google Cloud Platform* (GCP) — have incorporated specialized computer vision services, allowing developers to implement complex solutions in a scalable manner with reduced infrastructure effort. At the same time, embedded solutions, previously restricted by performance limitations and configuration complexity, have become more accessible, efficient, and easy to operate, thanks to the popularization of devices such as the Raspberry Pi, specialized NVIDIA boards, and the availability of optimized libraries for local inference (FOUNDATION, 2024; NVIDIA Corporation, 2025; ABDULHAQ; AHMED, 2025).

This scenario raises a recurring question among professionals and researchers in the field: which approach should be adopted to solve a real computer vision problem — a cloud-based architecture, an embedded solution, or a combination of both? The answer is not trivial, as it involves technical, economic, and operational variables that are frequently conflicting. Seemingly similar problems may require radically different architectures, depending on specific requirements such as latency, cost, robustness, and the level of technical specialization required.

Given that there is no single answer for all problems, an example problem was defined as the scope of the study. This chosen problem reflects a real pain point[1] for which one of the possible solutions is a classic architecture in the machine learning and computer vision segment: object detection, tracking, and counting.

Object detection, tracking, and counting are solutions that use computer vision to solve

[1] In this context, pain point refers to the concept used in market research and refers to an unsatisfied consumer need that can be solved through a product or service (PLATZER, 2018).

various real problems, such as monitoring people flow in public spaces, access control in restricted areas, parking space management, urban traffic analysis, logistics process automation, and productivity monitoring in industrial production lines, just to name a few (NAUMANN *et al.*, 2023; SINDAGI; PATEL, 2018; HEIMBERGER *et al.*, 2021).

Another real pain point that these solutions help mitigate, for example, is real estate evaluation based on local movement or traffic. In general, commercial and residential properties are evaluated based on subjective characteristics, such as perceived location, people flow estimates, facade visibility, and assumptions about the target audience profile, but without a foundation in data and facts that allow validating these conclusions. As a result, some properties are undervalued and others overvalued, which compromises the commercial exploitation potential of these properties (POURSAEED; MATERA; BELONGIE, 2017; CANNON; COLE, 2011).

A concrete example of the commercial relevance of local flow analysis is the startup SuaQuadra, which received an investment of R$ 21 million. The platform was created to facilitate the rental of commercial spaces, solving a classic problem: choosing a property without reliable data on people traffic and regional profile. With more than 2 thousand registered properties and a focus on geospatial data intelligence, the startup differentiates itself by offering information such as pedestrian flow, population density, and urban zoning — data that could also be generated via computer vision solutions — helping from small entrepreneurs to large franchises make more informed decisions. The valorization of this pain point — the lack of objective data for commercial property evaluation — reinforces the importance of solutions like the one proposed in this work, which uses computer vision to generate reliable data on local movement of people and vehicles (Startups Brasil, 2024).

For this problem, the objective of this work was defined as performing real-time video analysis and providing as a result the counting data of people and vehicles in a specific location. Counting problems through image analysis require two main computer vision stages: object detection (to locate people and vehicles) and object tracking (to follow the same object in a sequence of frames and assign it a unique identification).

Thus, an application was developed that uses different cloud computing services (Azure, AWS, and GCP) to perform the detection of people and vehicles in images. The same procedure is executed with *You Only Look Once* (YOLO) on an embedded system (Raspberry Pi), simulating edge computing. After detection, the objects are sent to a second processing stage dedicated to tracking and assigning unique identifications, enabling counting. This second stage is performed using *serverless* solutions on cloud computing platforms and, in the case of the Raspberry Pi, through local processing. In both cases, the *Deep Simple Online and Realtime Tracking* (DeepSORT) model is used for tracking.

During the experiments, various metrics are collected to measure quality, performance, and cost of each solution and provider, with the objective of identifying the most suitable approaches for different scenarios, as well as the strengths and weaknesses of each alternative.

The characteristics of the final solution bring together components from three major research areas: computer vision, cloud computing, and edge computing. Each of these areas is addressed in its own chapter, establishing the necessary knowledge for complete understanding of the application.

Chapter 2 — Computer Vision — discusses the definition of the field according to the main cloud computing providers and presents the fundamentals of this technology, highlighting some of its most relevant applications in contemporary society. Subsequently, the chapter delves into the topics of object detection and tracking, exploring different possible architectures for implementing these solutions, in addition to their main characteristics. Finally, the YOLO detection model is analyzed in detail, having been chosen for the detection stage in the embedded processing solution, including comparisons of some of its versions and DeepSORT.

In Chapters 3 and 4, the focus shifts from Computer Vision itself to how to deliver or develop these solutions. The chapter addresses the concept of cloud computing according to the main market providers, explaining its operation and highlighting the benefits. Although various applications are discussed, the focus falls on the computer vision services offered by Azure, AWS, and GCP. Additionally, the *container* execution solutions provided by each provider are detailed, essential for the tracking stage of the developed application.

As a counterpoint, cloud computing solutions have not always been available — and, over time, local alternatives have evolved to the point of competing in scenarios where independence, reduced cost, and low latency are crucial. Therefore, Chapter 4 — Edge and Embedded Computing — highlights the benefits of local processing versus cloud processing, especially in applications that require real-time response or offline operation. The discussions in this chapter focus on computer vision solutions, addressing the main hardware options and the most suitable deep learning models for this type of environment.

Thus, the initial chapters focus on the theoretical and practical foundation of the state of the art in each of the major fields - computer vision, cloud computing, and edge computing. From then on, the final chapters deal with the methodology, results, and conclusion of the research.

The methodology in Chapter 5 presents in detail the architecture of the application developed for this study. The technical decisions made at each stage are described, from the choice of tools to how the tests were conducted. The chapter also exposes the criteria used to measure performance, cost, and quality of solutions in different environments — cloud and edge —, as well as the data structure used to store and compare the results.

Chapter 6 — Results — consolidates the data obtained during the experiments, presenting objective comparisons between different execution scenarios. Metrics such as response time, resource consumption, counting accuracy, and operational cost are analyzed, allowing evaluation of the *trade-offs* involved in each approach. The presentation of results seeks to offer clear inputs for decision-making about which architecture proves most suitable for different application

contexts.

Finally, Chapter 7 — Conclusion — synthesizes the main learnings of this work, returning to the initial objectives and discussing to what extent they were achieved. The chapter also presents limitations identified throughout the development and proposes paths for future work, focusing on the evolution of the solution and possible new application scenarios.

CHAPTER

2

VISÃO COMPUTACIONAL

1826, Les Gras, France. Joseph Nicéphore Niépce, a man with a scientific mind but poor drawing skills, transformed his weakness into the first recorded photograph (Figure 1), initiating a revolution that would allow us to "immortalize" what the eyes see, record sequences of moving images, and carry countless memories wherever humans go (GERNSHEIM; GERNSHEIM, 1955).

Figure 1 – First photograph recorded by man.

Source: Center (2025).

Thus began a story about to complete 200 years and without which the central theme of this work might not exist or would be quite different.

But what is computer vision? How does it work? What is it used for?

2.1 What is computer vision?

There are different perspectives regarding this definition. For example:

> Computer vision is a field of artificial intelligence (AI) that uses machine learning and neural networks to teach computers and systems to extract meaningful information from digital images, videos, and other visual data — and to make recommendations or take actions when they identify defects or problems (IBM, 2025c).

> Computer vision is a field of computer science that focuses on enabling computers to identify and understand objects and people in images and videos. Like other types of AI, computer vision seeks to perform and automate tasks that replicate human capabilities. In this case, it seeks to mimic both how humans see and how they interpret what they see (MICROSOFT, 2025b).

> Computer vision is a technology that enables machines to automatically recognize images and describe them accurately and efficiently. Currently, computer systems have access to a large volume of images and videos from *smartphones*, traffic cameras, security systems, and other devices. Computer vision applications use artificial intelligence and machine learning (AI/ML) to process this data accurately, enabling object identification, facial recognition, classification, recommendation, monitoring, and detection (AMAZON, 2025d).

Such different definitions seem to contradict each other in some aspects. After all, would computer vision be "a field of artificial intelligence" (IBM, 2025c) or "a field of computer science" (MICROSOFT, 2025b)? Or would it still be a technology (AMAZON, 2025d)?

In a certain sense, all these statements seem adequate, since computer vision as technology presents itself as a tool that enables innovative solutions. At the same time, computer vision as a field of science (whether computer or artificial intelligence) allows for deepening and developments for the evolution of associated technology.

Despite the apparent contradictions and controversies, the different views presented about "computer vision" meet in some key concepts such as: the use of artificial intelligence as a means and the objective of providing practical solutions (recommendations, task automation, object identification, facial recognition, among others).

It is said that: "if artificial intelligence enables computers to think, computer vision enables computers to see, observe and understand" (MICROSOFT, 2025b).

As human beings, vision and understanding seem natural. But the human brain was, in fact, trained for years of context to associate characteristics captured by the eyes with ideas processed by neurons.

Similarly, computer vision as technology trains machines to perform such functions. However, many computer vision applications require high processing demand, in the sense that it would be humanly impossible to visually evaluate hundreds of thousands of images per minute. For example, studies identified that the human eye is capable of processing between 750 and 4,615 images per minute (POTTER *et al.*, 2014), while recent algorithms can achieve about 140,000 images per minute (NVIDIA, 2025).

This massive data processing has become viable through important advances in technology, whether through new hardware components that enable training powerful models with databases that were not previously possible, through the increase of such databases with proper annotations that allow quality training, or through new neural network architectures that propose intelligent solutions to leverage the results obtained.

In summary, computer vision is a powerful technology that mimics the human capacity to see and process images, with the objective of delivering value to those who use it.

2.2 How does computer vision work?

The functioning of this technology is based on machine learning and artificial intelligence applications.

During much of the history of computing, machines were programmed to respond to previously established commands, so that there was no talk of inherent intelligence, but rather of external intelligence capable of designing a logic architecture to produce certain results according to received inputs (TURING, 1950).

However, today, artificial intelligence is already a reality in human daily life and can be observed in applications such as:

- voice assistants that process speech, understand context, and respond appropriately;
- recommendation systems that offer personalized entertainment suggestions for each user;
- navigation systems that offer real-time route optimization;
- image analysis systems that detect disease risks;

just to name a few.

Artificial intelligence applied to image processing is possible through training on a set of images until the computer is capable of identifying patterns that describe certain objects.

As can be observed, the main difference in contrast with traditional programming is the ability of algorithms to learn on their own.

Multiple technologies were developed to assist in such training and pattern identification, of which two stand out with greater importance: deep learning and neural networks. But a model trained with such advanced technologies, but with a poor quality database, would still be a low-performance model. Therefore, it is necessary to highlight the importance that databases have for a good model.

2.2.1 Databases for computer vision model training

As important as the algorithm used by a computer vision system or the neural network architecture is the dataset used for training.

Image datasets are used for training models for image classification, facial recognition, object detection, and other computer vision-related applications. It is from these sets that algorithms learn to recognize the characteristics that will be sought when processing each image.

In general, these image sets can be obtained from 2 main sources: through manual collection and annotation, or through automatic collection via *web scraping*(ROH; HEO; WHANG, 2021)[1].

According to Bhujel *et al.* (2025), some image sets commonly used in computer vision applications are:

- Labelme: collection of images with ground truth labels for object detection and recognition;
- Pascal VOC: contains 20 object categories (vehicles, households, animals, airplane, bicycle, boat, bus, among others), where each image has pixel-level segmentation annotations, bounding box annotations, and object class annotations;
- ImageNet: organized according to WordNet hierarchy, where each node of the hierarchy is represented by hundreds and thousands of images;

However, an MIT study demonstrated that many datasets present labeling errors, so that the quality of the set becomes a concern. Such quality is evaluated according to 2 criteria: data quality (which includes duplicate, inconsistent, or missing data) and *metadata* quality (which includes problems with labeling, lack of complementary information, or incomplete information).

Among the risks associated with poor quality datasets used for training, one can cite the decline in model performance, lack of reliability in the model, which could produce unstable predictions, and even erroneous conclusions that lead to incorrect interpretations (which could affect even business-level decisions) (GONG *et al.*, 2023).

[1] *Web scraping*, or data scraping, is the automated extraction of data from websites. It is a technique used to collect information from web pages and convert them into data for analysis.

The conclusion is that image collections are essential for enabling computer vision solutions, and the quality of such collections directly impacts the quality of the solution to be developed.

2.2.2 Base technologies - deep learning and neural networks

Deep learning is a type of machine learning that uses neural networks composed of multiple layers of neurons that work in cooperation using mathematical calculations to automatically process different aspects of an image and gradually develop understanding.

More specifically, convolutional neural networks contribute mainly by "breaking down" images into pixels that receive labels and categorize the image. These pixels then go through a convolution, a mathematical operation, which results in a prediction or guess about the object being seen. Through the neural network, multiple convolutions are performed, and the accuracy of predictions or guesses is validated, until the predictions begin to be true, which indicates the point at which the algorithm was able to identify sufficient characteristics in its model that enable it to perform recognition activities in an image.

While convolutional neural networks are quite appropriate for image analysis, for video analysis a derivation of such networks is used, known as recurrent neural networks, which allow establishing the relationship between the sequences of images that form videos.

More practically, the image analysis process with computer vision begins through capture, which could happen through a camera, for example. Then, the image is sent to a device and used for processing. This device performs a search for patterns and compares these patterns with a library of already known patterns to determine if there is any compatibility between them. Based on the compatibility between patterns found in certain regions of the image and patterns already known in the library, it is estimated with a certain degree of certainty what the recognized object is.

There is, however, a need to distinguish computer vision from image processing. These technologies complement each other, but are essentially different. Image processing is the technology used to make adjustments and changes to the image itself, such as making an image sharper or blurred, changing colors, increasing or reducing resolution, among others. In many applications, image processing is used as a previous step to computer vision processing to increase detection quality. In other applications, computer vision is first used to identify images or parts of an image that are of interest, to then use image processing for adjustments and/or changes to the image. It is thus also established that computer vision is not image processing, but a complementary technology.

2.3 What is computer vision used for?

Many possibilities arise from the use of computer vision, so some examples of its applications are:

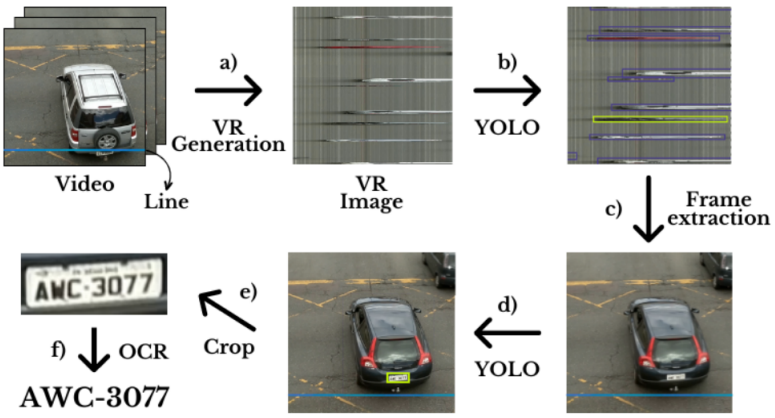
2.3.1 Text Extraction

Computer systems were initially designed to work with different types of data, so that text format data is processed differently from audio or image format data, for example.

This approach, although functional, introduces a limitation: the impossibility of processing texts provided as part of images or in image format. And such limitation made many solutions that could be developed for process automation unfeasible.

Through computer vision, however, it became possible for systems to acquire the ability to extract textual components from images.

Figure 2 – Steps of text extraction from vehicle plates using OCR and YOLO.



Source: Ribeiro and Hirata (2025).

One application is the extraction of vehicle plates from capturing images of traffic lanes. A camera continuously monitors vehicle flow, so that each image is processed by a system, which extracts specific characteristics and through an optical character recognition algorithm (*Optical Character Recognition* (OCR)) can determine character sequences, as demonstrated in figure Figure 2 (RIBEIRO; HIRATA, 2025).

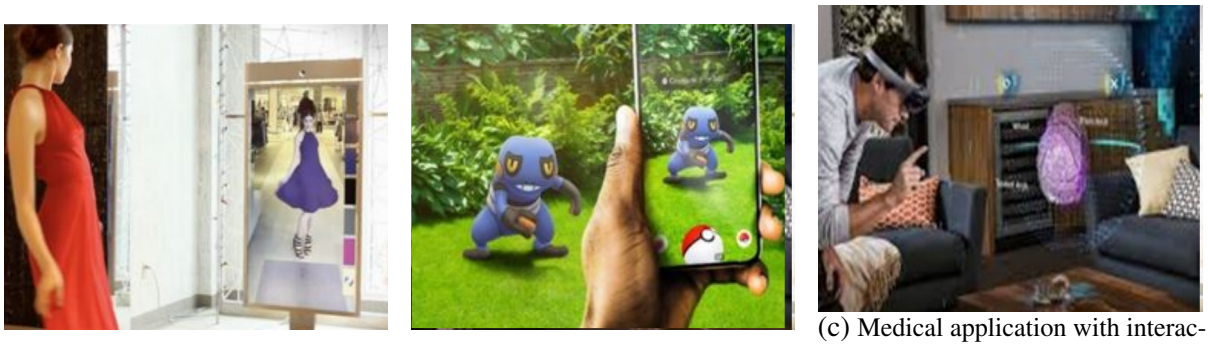
2.3.2 Augmented Reality

Augmented reality applications emerge as a way to complement the capture and processing of information performed by human eyes.

Using real-time capture, detection, and processing, information and objects are projected realistically overlaying the physically seen environment and allowing a more complete understanding of spaces, simulations, or even interaction with virtual environments.

Such characteristics allow applications such as those seen in Figure 3, where a person uses a smart mirror to test clothing options without the need to wear them, a doctor analyzes an organ from different perspectives, interacting with it, rotating, increasing its size to see more details, and even a young person who through their cell phone enjoys an augmented gaming experience by "bringing" virtual components to the real world (SINGH *et al.*, 2022).

Figure 3 – Examples of augmented reality in real life.



(a) Virtual clothing fitting room. (b) Augmented reality gaming. (c) Medical application with interaction.

Source: Singh *et al.* (2022).

2.3.3 Autonomous Cars

Although days have an average of the same 24 hours, it seems common to most people the feeling that days are getting shorter. This is a consequence of increased responsibilities for each person, leaving less and less time for tasks that bring pleasure.

Therefore, humans have continuously sought ways to improve the use of their time. In parallel, many spend significant time in traffic, time that is considered lost, since it cannot be used in any way.

Through the motivation to better use this time, as well as the objectives of increasing traffic safety and reducing traffic, large investments have been made in the development and improvement of autonomous cars, that is, vehicles with intelligence that allows displacement between two points without driver intervention or participation.

An autonomous vehicle, whether a car or even a drone, combines multiple technologies, one of them is radar or *Light Detection and Range* (LIDAR), which allows mapping the environment and measuring distances between objects.

However, while the vehicle module provides the ability to think or process information, and LIDAR provides the ability to know distances to obstacles (technologies that enable a certain level of autonomy), a contextual understanding of the environment is also necessary, not only

through detection of distance between objects, but also what meaning these objects provide for decision making.

Traffic signs, other vehicles, people, road lanes, traffic lights, among others, can even be detected by mapping sensors, but only through computer vision can their meaning be extracted.

An example is demonstrated by capturing images from autonomous vehicles under different weather conditions (DONG; CAPPUCCIO, 2024).

Figure 4 – Example of computer vision application in autonomous vehicle through different driving conditions.

Source: Dong and Cappuccio (2024).

2.3.4 Healthcare

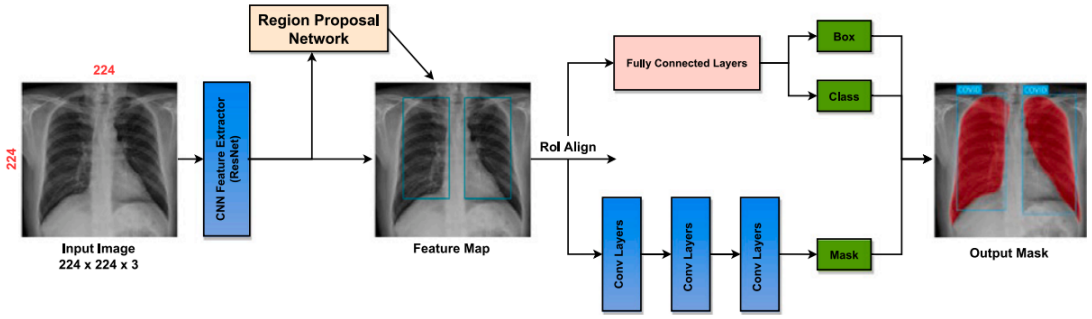
Human life expectancy has increased with the advancement of technologies in the health area. A large portion of these contributions have been possible through health devices used for problem identification and diagnosis in earlier stages, enabling treatments in initial phases of diseases and increasing recovery chances.

Computer vision is one of these technologies that have contributed to these advances.

For example, images captured by medical devices, whether X-rays, tomographies, or even photographs, can be analyzed to provide help to doctors in problem identification and diagnosis.

In Figure 5 it is possible to see a chest X-ray, which is used as input in a computer vision algorithm for feature detection, and then by a second algorithm for COVID detection (KABIR *et al.*, 2025).

Figure 5 – Example of computer vision in use in healthcare through chest X-ray processed for COVID detection.



Source: Kabir *et al.* (2025).

These 4 examples presented demonstrate some of the areas in which computer vision has been applied at scale and demonstrate well how different computer vision applications can be used to create distinct solutions. In this way, the great importance, applicability, and scalability of solutions that use computer vision in the current era is established.

The example applications presented, despite representing solutions to different problems, are based on the use of computer vision techniques that are common to most, such as object detection, feature extraction, depth perception, semantic segmentation, among others.

Among such techniques, two in particular have great relevance for the development of this work: object detection and object tracking.

2.4 Computer vision techniques: object detection and tracking

Computer vision encompasses various techniques that enable automatic analysis and interpretation of images and videos by computational systems. Among these techniques, object detection and tracking stand out for their relevance in applications ranging from security to traffic analysis and industrial automation. Detection focuses on identifying and locating specific objects in images or video sequences, while tracking aims to follow these objects over time, ensuring dynamic understanding of the environment. This section addresses the main methods and approaches for detection and tracking, exploring from classical algorithms to techniques based on deep learning.

2.4.1 Object detection

Object detection is the computer vision technique used to locate instances of objects in images or videos, with the objective of replicating human intelligence that allows recognizing

and locating objects of interest in images captured by the eyes in a matter of moments (IBM, 2025a).

This technique is important because without it many solutions that have become part of daily life would not be possible, such as visual inspection solutions, robotics, medical imaging, video surveillance, and content-based image retrieval, just to name a few examples.

Object detection is similar to another technique widely used in computer vision and known as image segmentation. However, image segmentation aims to be more precise in detection, in the sense that while object detection identifies through frames the region of the object, image segmentation demarcates at the pixel level, as demonstrated in Figure 6.

Figure 6 – Difference between object detection (left) and image segmentation (right).

Source: Hassan and Sabha (2023).

But how does object detection work?

2.4.2 Object detection implementation

Object detection is a task that involves determining the specific position of objects using a bounding box, and classifying objects, which defines which category the detected object belongs to.

This process begins with the way a computer reads, visualizes, or represents an image. When digitized, an image is subjected to 2 main procedures called sampling and quantization[2], which convert the image into a discrete structure of pixels, in which the x and y axes are used to identify the pixel position, and a tuple[3] of values is used to characterize the data associated with that point, such as color (possibly in RGB format) and transparency.

[2] Quantization is the process of reducing the precision of a digital signal, usually from a higher precision format to a lower precision format. This technique is widely used in various fields, including signal processing, data compression, and machine learning. In image analysis, quantization involves, for example, reducing the number of colors or intensity levels in an image (IBM Corporation, 2025b).

[3] A tuple is an ordered set of values represented by a vector. This type of data structure is useful for representing data models and can vary between one programming language and another (Python Software Foundation, 2025).

From this pixel matrix, the computer can segment the image according to similarities between pixels close to each other.

In the dataset used for training, in addition to images, labels are available that describe the areas of each image that contain objects and which objects are contained there. Allowing the model to learn the characteristics that determine certain objects.

From there, when it receives an input image, the model searches for regions similar to those already seen and abstracted from the training image collection. Thus, object detection presents itself as a special form of pattern recognition algorithm, since the objects themselves are not recognized, but rather the set of properties or patterns that define such objects.

Finally, annotations are made on the image to delimit the identified objects along with their classes.

These steps are a high-level and simplified view that help in general understanding, but there are different ways to implement such steps, each with its advantages and disadvantages, some of which are the following (MathWorks, Inc., 2025):

1. object detection using deep learning with pre-trained detectors;
2. object detection using deep learning with custom detectors;
3. object detection using machine learning with aggregated channel features;
4. object detection using machine learning with support vector machine classification.

2.4.2.1 Object detection using deep learning with pre-trained detectors

Several deep learning object detectors are trained on large datasets and can detect common objects, such as people, vehicles, or image texts, without requiring additional training.

Pre-trained detectors are models that have been previously trained and then made available for public or licensed use.

Some of these models are YOLO, *Vision Transformer* (ViT), *Single Shot MultiBox Detector* (SSD), Faster and Mask *Region-based Convolutional Neural Network* (R-CNN), among others (JIAO *et al.*, 2019).

R-CNN models are two-stage models that use a method called "region proposal" to generate 2000 prediction regions per image. From there, these regions are used as input to different neural networks for feature extraction and classification (in distinct stages). Each region is ranked according to classification confidence. If there is a large overlap between two regions, then these will be discarded. The main disadvantage of these models is being slow and expensive, but this disadvantage was reduced after some advances with Faster-RCNN (REN *et al.*, 2016).

The YOLO model or "you only look once" is known for its speed, enabling real-time image processing. Such performance is a result of its architecture that differs from other models that use detection methods based on regions of interest (such as R-CNN). Region of interest-based detection first needs to determine potential object locations in images, to then perform classification. While in YOLO architecture, these two steps are performed simultaneously (single-shot).

The biggest challenge for this model is processing small objects or objects that are very close to each other, but advances have allowed mitigating this challenge and providing satisfactory results.

The YOLO architecture will be addressed in more detail in section 2.5.

Detectron2 is actually a model library developed by Facebook's artificial intelligence research team, using the PyTorch framework. This library contains models trained from different image sets, such as *Common Object in Context* (COCO) and ImageNet. Each of the models in this library was trained from different architectures, such as Faster R-CNN, Mask R-CNN, and RetinaNet. But the great advantage of using Detectron2 is its flexibility and modularity that allow testing different variations according to business needs (MERZ *et al.*, 2023).

The Vision Transformer, or ViT, is a model for image classification that employs an architecture similar to a transformer over image patches (DOSOVITSKIY *et al.*, 2021).

A transformer in machine learning is a deep learning model that uses attention mechanisms[4], differentially weighting the significance of each part of the input data sequence.

Transformers offer a generic learning approach that can be applied to different types of data. In computer vision, they achieve comparable or superior accuracy to convolutional networks (CNNs), with greater efficiency in parameter usage.

ViT outperforms CNNs in various scenarios, requiring fewer computational resources in pre-training. However, it depends more on regularization and data augmentation (AugReg) techniques, especially when trained with smaller datasets.

The SSD model was introduced in 2016 and stood out for an efficient single-shot approach[5] (LIU *et al.*, 2016), where a single neural network performs detection and classification simultaneously, this approach is the same used by YOLO.

[4] An attention mechanism is a machine learning technique that directs deep learning models to prioritize the most relevant parts of input data. The innovation in attention mechanisms enabled the transformer architecture that produced modern LLMs (Large Language Models) that power popular applications like ChatGPT.

[5] The single-shot technique offers greater processing speed, being more appropriate for use in applications where processing power is more limited or where there is high demand for images to be processed per unit time (e.g., real-time applications). However, the opposite alternative, two-stage detection, in which object localization and classification are performed independently, generally presents results with greater precision (PARAB *et al.*, 2022).

The design of this model's architecture uses a convolutional neural network to extract image characteristics during training, and then additional layers make predictions for different object sizes, making detection more robust in this sense.

Despite having stood out for a time, the SSD model is already in disuse with the emergence of more modern technologies, such as more recent versions of YOLO and approaches that use vision-transformers, but being a lightweight and fast model, it is still applicable for edge processing or embedded systems.

2.4.2.2 Object detection using deep learning with custom detectors

Pre-trained models are very practical since they offer quick solutions to common problems. But some applications require custom models for specific problems.

For example, COVID was not known until 2020, when this virus spread and became a global pandemic. In this way, there were no deep learning models capable of performing quality facial recognition on people wearing masks. This was such a specific application that existing image sets did not foresee, and that therefore, pre-trained models did not know[6] (ZHANG *et al.*, 2025).

To solve this problem, image collections of people wearing masks were created and then pre-trained models were customized to additionally acquire the ability to recognize faces of people using different types of masks.

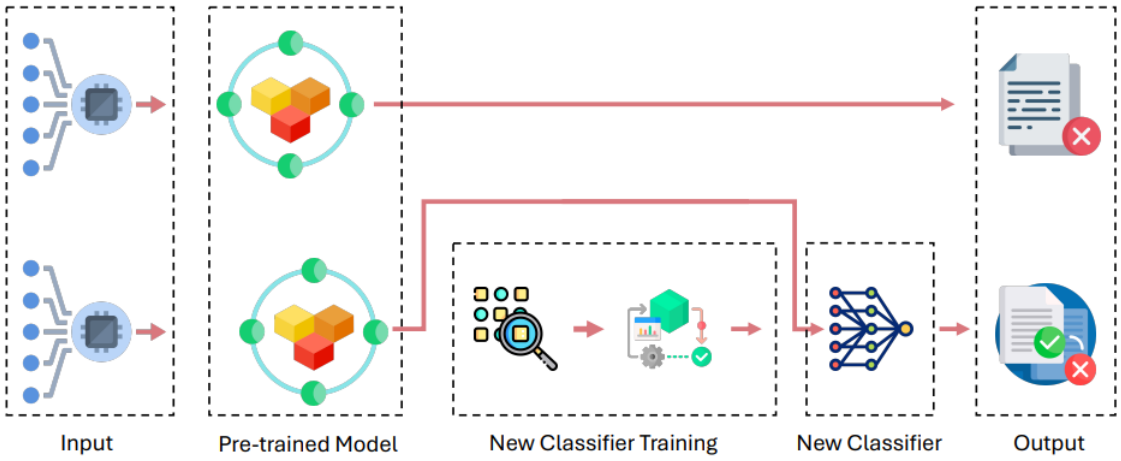
In this sense, 2 approaches can be used. Creating a model from scratch, using an existing image collection and adding a complementary image collection to meet the personalized requirement, or refining the pre-existing model through transfer learning[7] (PANDA *et al.*, 2024).

Figure 7 exemplifies the transfer learning process, where an additional step is performed in sequence to classification by the pre-trained model.

[6] For comparison purposes, Zhang's research showed that from 2018 and 2019 to the period from 2020 to 2023, the number of publications of works related to facial recognition with masks increased from 594 to 6484.

[7] Transfer learning is a machine learning technique where a model pre-trained on a large dataset is reused and adjusted for a new task with a smaller dataset. This technique is widely used as it reduces training time, eliminating the need to train the model from scratch, but preserving its useful characteristics.

Figure 7 – Example of transfer learning pipeline.



Source: Panda *et al.* (2024).

2.4.2.3 *Object detection using machine learning with aggregated channel features*

The techniques presented previously make use of deep learning, which extends through neural networks.

However, such techniques have not always existed, so many computer vision solutions were developed using machine learning. These solutions also comprise models for object detection.

Some object detection models were developed with the aid of a machine learning technique known as aggregated channel features (*Aggregated Channel Features* (ACF)) (BASTIAN; C.V., 2019), which uses multiple features extracted from the image to feed a classifier. Among the extracted features are: gray levels, directional gradients, gradient magnitudes, among others.

Despite being an old and limited approach, since it has lower precision, lower generalization capacity, and depends on manually extracted features, there are still some contexts where its use can make sense, such as training from smaller datasets and lower processing power requirement.

2.4.2.4 *Object detection using machine learning with support vector machine classification*

Like ACF, the support vector machine classification approach is also legacy and depends on feature extraction that is then used as classifier input.

The support vector machine (*Support Vector Machine* (SVM)) is the name of the classifier used in this case, a machine learning classifier that is not designed to work with images, so feature extraction is an important preliminary step (OSUNA; FREUND; GIROSIT, 1997).

Among the extracted features one can cite: histograms of oriented gradients (*Histogram of Oriented Gradients* (HOG)) and wavelets for interest point detection.

Like ACF, approaches that use SVM are lighter than the neural networks used in deep learning approaches. But despite lightness contributing to its use in applications where processing power is limited, one should consider the challenges of feature extraction, reduced precision, and sensitivity to noise and class imbalance, which harm the quality of results obtained.

2.4.2.5 Deep learning or machine learning - how to choose?

As presented in the previous subsections, approaches for object detection can be classified into two main groups:

- Deep learning, with detectors that learn features directly from data (e.g., YOLO, Faster R-CNN); or
- Machine learning, which depends on manual features (ACF, HOG + SVM, techniques like Viola–Jones).

The selection between these techniques is not arbitrary, but depends on the application context, which according to Karypidis *et al.* (2022), should consider:

- Type of object and scenario complexity: small, overlapping objects or in varied environments tend to benefit from deep networks, which automatically learn discriminative representations;
- Amount of labeled data: deep learning requires large volumes to avoid overfitting[8], so traditional methods may be superior in environments with limited data;
- Computational resources and real-time: deep learning models are heavier and generally require GPUs, while classical machine learning techniques (HOG + SVM) are lighter and more suitable for embedded devices.
- Availability of pre-trained models: detectors like YOLO and Faster R-CNN can accelerate prototypes, but may require fine-tuning[9] when the domain scope differs from the original dataset.

In summary: there is no universally superior approach. The ideal choice is one that balances performance, data and resource requirements, and suitability to the specific problem.

[8] Overfitting is a characteristic that manifests when an algorithm adapts excessively or even precisely to training data, leading to a model that cannot make accurate predictions or conclusions with data other than training data (IBM Corporation, 2025c).

[9] Fine-tuning in machine learning is the process of adapting a pre-trained model for specific tasks or use cases (IBM Corporation, 2025a).

2.4.3 Object tracking

Object tracking is a computer vision technique similar to object detection, but that expands its capabilities, since in addition to locating and classifying the object, object tracking is also capable of following this object throughout a sequence of frames in a video (or sequence of images).

The object tracking technique requires object detection as a preliminary step, which can be performed through the models studied in the previous section.

From object detection, object tracking models then need to follow the object in the next frame. Two different strategies allow achieving this result.

One option is to try to follow the object's movement, through the use of algorithms that predict the displacement the object will make, one of these algorithms is called optical flow (KALE; PAWAR; DHULEKAR, 2015).

The optical flow algorithm calculates movement between two frames at different times, pixel by pixel, so that it is possible to clearly separate the background from moving objects and determine how much each object has moved.

Another option is to try to identify characteristics in the object that represent it uniquely and then, in the next frames, identify if there is any object with similar characteristics.

Feature identification makes use of descriptors. Descriptor algorithms are programmed to find certain characteristics in an image, ignoring everything else. There are 2 types of descriptors that are very common: descriptors based on local gradient and descriptors based on image intensity (LIU *et al.*, 2021).

Among gradient-based descriptors one can cite some like *Scale Invariant Feature Transform* (SIFT) and HOG. Among image intensity-based descriptors, one can cite some like *Oriented FAST and Rotated BRIEF* (ORB) and *Fast Retina Keypoint* (FREAK).

Through such approaches, object tracking enables more robust computer vision solutions, such as pedestrian, vehicle, and obstacle tracking, security monitoring, and sports analysis.

2.5 YOLO (*You Only Look Once)*

The YOLO model or "you only look once" was introduced in 2016 (REDMON *et al.*, 2016), and its name originates from its single-shot approach or single-stage detection approach. This work proposed an innovative approach for the object detection task, in contrast to the multi-stage detectors predominant until then, such as R-CNN. YOLO stood out for transforming object detection into a unified regression problem, enabling real-time detection with high speed.

The main objective of YOLO models is to perform object detection in images and videos with high performance and low latency. Its differential lies in the ability to process a complete

image in a single neural network inference, which guarantees considerable agility compared to other traditional detectors.

Currently, the development and maintenance of the most popular versions of YOLO models are conducted by different groups. From YOLOv5, Ultralytics assumed protagonism in model development and dissemination, providing official repositories and documentation (JOCHER *et al.*, 2020). More recent versions, such as YOLOv8 and YOLOv11, were also developed and maintained by this same organization, with continuous improvements aimed at ease of use, performance, and support for multiple tasks, such as segmentation and pose estimation (ULTRALYTICS, 2023; ULTRALYTICS, 2024c).

The YOLO architecture is based on convolutional neural networks (CNNs). The general functioning involves dividing the input image into a grid and, for each cell of this grid, the model predicts:

- Bounding boxes (object delimiting rectangles),
- Confidence of object presence,
- Classes to which the object may belong.

Researchers and developers have worked continuously to improve this model through elaborate strategies, among which one can cite the adoption of optimized backbone architectures, such as *Cross Stage Partial Network* (CSPNet), and modern techniques such as robust data augmentation, use of optimized anchors, neck with *Path Aggregation Network* (PANet) for better feature fusion, and a head dedicated to object prediction (Ultralytics, 2025b).

These modifications made YOLO more efficient, scalable, and adaptable to different tasks.

What enables the high detection speed of this model is the single-shot architecture that performs object detection in a single pass of the image through the network, unlike approaches based on multiple stages, such as R-CNN and its variants. While multi-stage models first generate region proposals to then classify them, YOLO divides the image into a fixed grid and simultaneously predicts object classes and coordinates of their bounding boxes. This characteristic ensures extremely high inference speed, enabling real-time applications — one of the great differentials of the model since its first proposal (REDMON *et al.*, 2016).

The backbone component is responsible for extracting the main features from images. From more recent versions, such as YOLOv4 and its evolutions, the model began adopting CSPNet as backbone (WANG *et al.*, 2019). CSPNet was proposed with the objective of improving gradient flow in the network, reducing information duplication and computational cost, without significant performance loss. This improvement is obtained by dividing feature maps into two

parts: one goes through deeper transformations, while the other is connected directly to the output, promoting better use of information.

Another important characteristic of more recent YOLO versions is the intensive use of data augmentation, through the application of artificial variations in training images — such as rotations, scaling, geometric distortions, brightness or color changes, among others — with the objective of increasing model robustness and its generalization capacity (SHORTEN; KHOSHGOFTAAR, 2019). This strategy proves fundamental for object detection in real scenarios, where lighting, angles, and noise vary considerably.

Also, the use of optimized anchors (reference boxes with pre-defined dimensions) to facilitate regression of final bounding boxes contributes to greater model precision, since appropriate anchor selection — normally performed through clustering algorithms (k-means clustering) on the training set allows the model to better adapt to the distribution of sizes and proportions of objects present in the data (BONILLA *et al.*, 2023).

Since YOLO is a deep learning model, the way information is extracted and communicated between network layers plays a fundamental role. Therefore, the use of PANet architecture contributes to improving communication between shallow and deep network layers, enabling aggregation of detailed (or low-level) and contextual (or high-level) information (LIU *et al.*, 2018). This architecture results in more precise detection in complex images or with objects of varying sizes.

Finally, YOLO's head is the part of the network responsible for effective prediction of boxes and classes of detected objects. Each grid cell, after processing by previous steps, generates box coordinates, predicted classes, and prediction confidence. Adequate design of this head is essential to maximize precision and minimize detection errors.

These are the steps and technologies that allow YOLO to offer results with greater precision and speed.

The variety of model sizes also contributes to YOLO's popularity. Among its many versions, the current one is v11, with v12 in development, there are models of various sizes, such as nano, small, medium, large, and XLarge.

This variety of sizes allows use with devices of different processing powers, from embedded devices to large servers.

Among the disadvantages that can be cited of this model line are lower precision compared to multi-stage models and limitation in detecting very small, nearby, or overlapping objects. In contexts where YOLO does not perform well, other architectures, such as those described in previous sections (Detectron2, vision transformer, SSD, machine learning with aggregated channel features, machine learning with support vector machine classification, among others) may be more adequate.

Table 1 – Technical comparison of characteristics or metrics between YOLOv5, YOLOv8, and YOLOv11 models.

Characteristic	**YOLOv5 (2020)**	**YOLOv8 (2023)**	**YOLOv11 (2024)**
Precision ($mAP_{50\text{-}95}$)	42.1% (v5n) 56.8% (v5x)	37.3% (v8n) 53.9% (v8x)	39.5% (v11n) 54.7% (v11x)
CPU Speed	211.0 ms (v5n) 2436.5 ms (v5x)	80.4 ms (v8n) 479.1 ms (v8x)	56.1 ms (v11n) 462.8 ms (v11x)
GPU Speed	1.83 ms (v5n) 8.98 ms (v5x)	0.99 ms (v8n) 3.53 ms (v8x)	1.5 ms (v11n) 11.3 ms (v11x)
Parameters (millions)	4.3 (v5n) 155.4 (v5x)	3.2 (v8n) 68.2 (v8x)	2.6 (v11n) 56.9 (v11x)
FLOPs (billions)	7.8 (v5n) 250.7 (v5x)	8.7 (v8n) 257.8 (v8x)	6.5 (v11n) 194.9 (v11x)
Embedded Devices	Suitable	Optimized	Highly optimized

Source: Ultralytics (2024a).

Various metrics can be used to evaluate a model, among which the main ones are *mean Average Precision* (mAPval 50-95), CPU speed, GPU speed, number of model parameters, and number of floating-point operations required for one inference (*Floating Point Operations* (FLOPs)) (Ultralytics, 2025a).

The mAPval 50-95 metric measures the model's average precision considering different intersection thresholds between the predicted box and the real box, varying from 50% to 95% overlap. The higher this value, the better the model's performance in the object detection task. Values above 70% are generally considered excellent, although this number may vary depending on the complexity of the dataset used. In Table 1 the COCO dataset was used (ULTRALYTICS, 2024a). For YOLO, since the objective is to provide real-time detections, prioritizing speed over quality, high mAP values are not expected, they are generally around 50%.

In addition to precision, it is important to evaluate performance in terms of inference speed. The CPU speed metric indicates the average time, in milliseconds, that the model takes to make a prediction when executed on CPU. Values below 100 milliseconds are considered fast, while values above 300 ms can be problematic in real-time applications. The GPU speed metric measures inference time using a GPU. In this case, times below 5 ms indicate exceptional performance, especially in applications that demand high speed.

Two other important metrics are related to model complexity. The number of parameters represents the number of model parameters, in millions. Models with fewer than 10 million parameters are considered extremely lightweight and more suitable for execution on devices with limited resources. Finally, the FLOPs metric indicates the number of floating-point operations required to perform one inference, in billions. This value is directly related to the model's computational cost: the lower the number of FLOPs, the more efficient execution tends to be,

especially in environments with hardware restrictions.

These metrics, when analyzed together, allow a balanced view between precision and computational performance, helping in choosing the most suitable model according to application requirements.

Table 1 presents a technical comparison between YOLOv5, YOLOv8, and YOLOv11 versions. In terms of precision ($mAP_{50\text{-}95}$), YOLOv5 varies from 42.1% (v5n version) to 56.8% (v5x), YOLOv8 presents from 37.3% (v8n) to 53.9% (v8x), while YOLOv11 achieves from 39.5% (v11n) to 54.7% (v11x).

In CPU inference speed, YOLOv5 is the slowest, with 211.0 ms (v5n) and 2436.5 ms (v5x). YOLOv8 significantly improves this performance, with 80.4 ms (v8n) and 479.1 ms (v8x). YOLOv11 presents the best results, with 56.1 ms (v11n) and 462.8 ms (v11x).

Regarding GPU speed, YOLOv8 is the fastest in smaller versions, with 0.99 ms (v8n), followed by YOLOv5 with 1.83 ms (v5n) and YOLOv11 with 1.5 ms (v11n). In larger versions, YOLOv8 maintains advantage with 3.53 ms (v8x), against 8.98 ms (v5x) and 11.3 ms (v11x).

Regarding the number of parameters, a reduction trend is observed throughout versions. YOLOv5 has 4.3 million (v5n) and 155.4 million (v5x), YOLOv8 reduces to 3.2 million (v8n) and 68.2 million (v8x), while YOLOv11 presents 2.6 million (v11n) and 56.9 million (v11x).

In the number of floating-point operations (FLOPs), YOLOv5 varies from 7.8 billion (v5n) to 250.7 billion (v5x). YOLOv8 presents from 8.7 billion (v8n) to 257.8 billion (v8x). YOLOv11 is the most efficient, with 6.5 billion (v11n) and 194.9 billion (v11x).

Finally, regarding suitability for embedded devices, YOLOv5 is only suitable, YOLOv8 is optimized, while YOLOv11 is highly optimized, evidencing its focus on applications with hardware restrictions.

2.6 DeepSORT (*Deep Simple Online and Realtime Tracking*)

DeepSORT is one of the most popular algorithms for multiple object tracking (*Multiple Object Tracking* (MOT)) in videos, being widely used in applications that require unique identification and continuous tracking of objects over time (WOJKE; BEWLEY; PAULUS, 2017). DeepSORT was proposed as an extension of the *Simple Online and Realtime Tracking* (SORT) algorithm, incorporating deep learning techniques to improve association of detections between consecutive frames.

Unlike the original SORT, which is based only on geometric information (position and velocity of objects), DeepSORT incorporates a visual feature extraction component through a

pre-trained CNN to generate discriminative appearance embeddings[10].

Thus, the main innovation of DeepSORT over the original SORT lies in incorporating deep visual characteristics, which allow distinguishing visually similar objects and maintaining identity even after occlusions or abrupt trajectory changes. This results in a significant reduction of identity switch errors (ID switches) and track fragmentations (BEWLEY *et al.*, 2016).

DeepSORT's architecture is composed of two main components: the tracker based on Kalman filter and detection association using deep learning. The Kalman filter is responsible for predicting the future position of tracked objects, taking into account movement history. Detection association between detections and existing tracks is performed through a similarity metric that combines spatial information (position and size of bounding box) and visual characteristics extracted by a convolutional neural network trained for re-identification (ReID) (WOJKE; BEWLEY; PAULUS, 2017).

DeepSORT's functioning can be summarized in the following steps:

- Object detection in each frame, normally using models like YOLO, Faster R-CNN, or SSD;
- Extraction of visual characteristics (embeddings) from detected regions, through a ReID network;
- Prediction of tracked object positions using Kalman filter;
- Association between detections and existing tracks, using the Hungarian algorithm and a cost metric that combines Mahalanobis distance (for position) and cosine distance (for appearance) (WOJKE; BEWLEY; PAULUS, 2017);
- Track updating and assignment of unique persistent identifiers.

DeepSORT's functioning depends on integration with an external object detector for the first step, such as YOLO or SSD. The detector is responsible for providing frame-by-frame detections. DeepSORT then acts in the subsequent step, performing tracking-by-detection, where tracking is conducted based on received detections and not directly on images.

In this way, DeepSORT allows integration with different object detectors and ReID networks, which facilitates its adaptation to different domains and precision or speed requirements. In practical applications, the choice of detector and ReID architecture can be adjusted according to scenario complexity and computational restrictions.

Among the main metrics used to evaluate multiple object tracking algorithms, the following stand out: *Multiple Object Tracking Accuracy* (MOTA), *Multiple Object Tracking*

[10] A numerical vector that represents the visual appearance of a detected object in an image. This vector is generated by a pre-trained CNN, which extracts discriminative visual characteristics from the image crop where the object appears.

Precision (MOTP), number of ID switches, and processing time per frame. DeepSORT presents competitive performance in these metrics, being capable of operating in real-time on modern hardware, especially when combined with fast detectors like YOLO (WOJKE; BEWLEY; PAULUS, 2017; CIAPARRONE *et al.*, 2020).

Despite its advantages, DeepSORT may present limitations in scenarios with great variation in object appearance, imprecise detections, or highly dynamic environments. In these cases, more recent approaches, based on end-to-end models or with temporal attention, may be considered.

CHAPTER

3

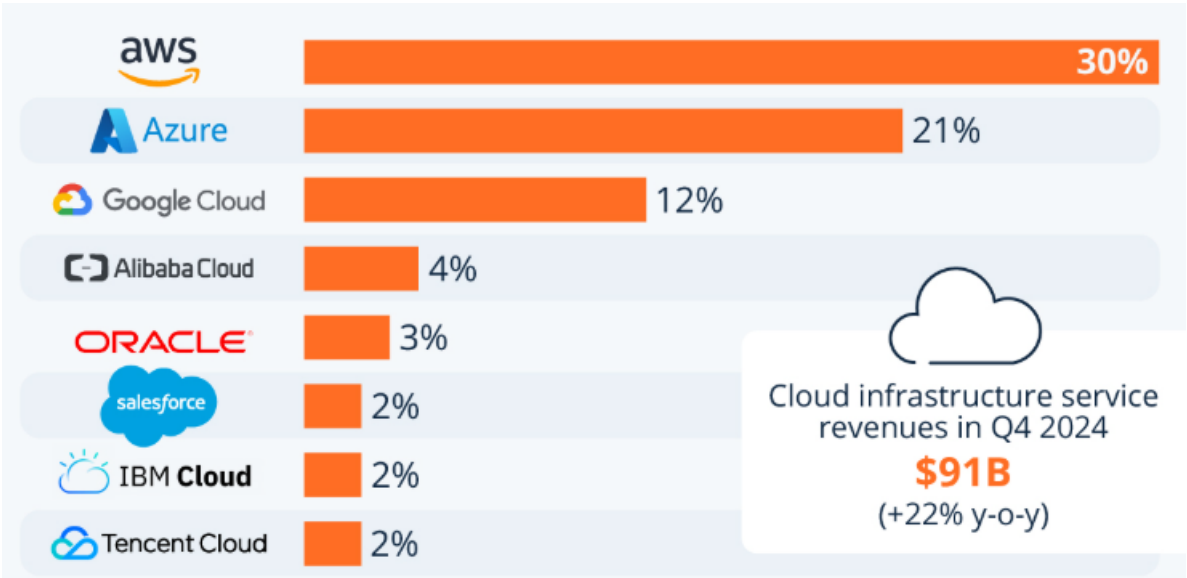
COMPUTAÇÃO EM NUVEM

Cloud computing is a technology that has become increasingly popular every day. But, unlike computer vision, addressed previously, cloud computing is in many cases offered as a service by a provider.

Among the most well-known cloud computing providers, three can be mentioned: Microsoft, Amazon, and Google.

These computing giants dominate the market with about 63% of total segment revenue, with Amazon holding 30%, Microsoft with 21%, and Google with 12%, as observed in Figure 6 (Statista, 2025).

Figure 8 – Cloud computing market share



Source: Statista (2025).

These providers compete in a market worth almost R$ 1,891,000,000,000 (yes, almost 2

trillion reais)[1]. As impressive as the market size is the accelerated growth of the same. In 2022, the segment generated 1.36 trillion reais in revenue, a value that grew approximately 14% in 2023, reaching 1.55 trillion reais. (Synergy Research Group, 2025).

This accelerated growth in recent years alone highlights the importance that cloud computing has had in the world economy.

But what is cloud computing? How does it work? What is it used for?

3.1 What is cloud computing?

To begin this discussion, it is convenient to analyze how the technology giants that compete for this market self-define this term.

> Cloud computing is the on-demand availability of computer system resources (such as storage and infrastructure) as services over the internet. It eliminates the need for individuals and businesses to manage physical resources on their own, allowing them to pay only for what they use (GOOGLE, 2025).

> Cloud computing is the delivery of computing services — including servers, storage, databases, networking, software, analytics, and intelligence — over the internet ("the cloud") to offer faster innovation, flexible resources, and economies of scale. You typically pay only for cloud services you use, helping lower your operating costs, run your infrastructure more efficiently, and scale as your business needs change (MICROSOFT, 2025a).

> Cloud computing is the on-demand delivery of IT resources over the internet with pay-as-you-go pricing. Instead of buying, owning, and maintaining physical data centers and servers, you can access technology services, such as computing power, storage, and databases, on an as-needed basis from a cloud provider like Amazon Web Services (AWS) (AMAZON, 2025c).

Despite competition for the market, Amazon, Microsoft, and Google providers do not compete when it comes to defining what cloud computing is. The above citations allow us to conclude that cloud computing is a service offered by providers.

One of the characteristics of this service is its predominantly *pay-as-you-go* billing model, where the customer pays only for what they used.

[1] If the cloud computing segment were a country, its GDP (gross domestic product) would be among the 50 largest in the world, surpassing countries like Chile, Portugal, and New Zealand

Other characteristics of the cloud computing service[2] are internet access and the variability of available resources, among which can be mentioned: servers, databases, software, among others.

Cloud computing emerges as an alternative to local data processing and storage. For example, consider a company that develops *websites*. Each site is formed by a set of files. For these sites to be accessed, their files need to be available somewhere. This company might then need many servers to host the files of the *sites* of each of its clients. In traditional computing, these servers would be property and responsibility of this website development company. Being part of their scope the regular maintenance, monitoring, and expansion of servers, just to name a few examples.

On the other hand, in cloud computing, these servers would exist in the same way and with the same objective, but now, the ownership and responsibility for these servers belongs to a third-party company, a company called a cloud service provider (*Cloud Service Provider* (CSP)).

3.1.1 Cloud computing service categories

This is just one example of a service that a provider offers. In fact, the main services can be divided into some categories:

Infrastructure as a Service (*Infrastructure as a Service* (IaaS)): is the most basic cloud computing service and consists of renting IT infrastructure, including servers and virtual machines (VMs), storage, networks, and operating systems.

Platform as a Service (*Platform as a Service* (PaaS)): is the service that provides an on-demand environment for developing, testing, delivering, and managing software applications, to facilitate rapid creation of web or mobile applications, without the need to configure or manage the infrastructure necessary for this development. It is typically built from *containers*, a virtualized computing model.

Software as a Service (*Software as a Service* (SaaS)): is a service that offers software applications over the Internet, on demand and typically by subscription.

***Serverless computing*:** this service also called serverless computing focuses on building applications without the need to manage infrastructure, providing highly scalable and event-driven alternatives, which use resources only when a specific function or trigger occurs.

This different range of services contributes to making cloud computing an attractive solution for different audiences. Knowing the differences between these models is important for choosing appropriate solutions according to the requirements of each project.

[2] The "cloud" refers to a set of powerful servers housed in large *data centers* and that can be accessed through the internet.

3.1.2 Types of cloud

In addition to knowledge of service delivery models, it is also necessary to understand that there are different types of cloud, in this sense, the main ones are:

Public cloud: is a type of cloud in which resources are made available to users through the public internet, either for free or according to subscription-based or pay-per-use pricing models. In this model, different customers gather and share the cloud provider's resources.

Private cloud: is a type of cloud where all cloud infrastructure and computing resources are dedicated to just one customer, combining cloud computing benefits like elasticity, scalability, and ease of service delivery, with the access control, security, and resource customization of on-premises infrastructure. Many companies choose a private cloud instead of a public cloud environment to meet regulatory compliance requirements. Large entities, such as government agencies, healthcare organizations, and financial institutions, often opt for private cloud configurations for workloads that handle confidential documents, personally identifiable information, intellectual property, medical records, financial data, or other sensitive data.

Hybrid cloud: is a combination of public cloud, private cloud, and on-premises environments in a single, flexible infrastructure. A survey showed that 77% of companies and IT professionals have adopted a hybrid approach (IBM, 2025b).

The different types of cloud, public, private, and hybrid, exist to give flexibility to users and allow the best choice according to their needs.

3.1.3 Benefits of cloud computing

The business model based on cloud computing has some advantages for companies that adopt it. (IBM, 2025b):

Cost-effectiveness: Cloud computing requires a much lower initial investment, since there are no expenses for purchasing materials (*hardware* and *software*, for example), installation and configuration, and even some operational costs are avoided such as specialized labor costs and maintenance. In this way, the customer only pays for resources that were effectively used.

Speed and agility: Cloud computing providers offer more than databases and servers, but also complete solutions that speed up technology implementation by companies. In some circumstances, access to applications can be made available in a matter of minutes.

Scalability: Cloud computing provides elasticity and self-service provisioning, so instead of buying excess capacity that sits unused during periods of low demand, customers can increase and decrease resource allocation in response to demand spikes and drops.

Lower latency: Latency is the response time when a request is made to a server. A request made by a user in Africa to a server located in Brazil has higher latency or response time than a request made by a user in Brazil, for example, in São Paulo, to a server also in São Paulo, Brazil. Cloud computing providers offer the possibility of allocating servers in different regions of the world and rules can be created to direct each user to the server from which they are closest, reducing response time.

Security: Cloud providers also offer a comprehensive set of policies, technologies, and controls that strengthen overall security and protect data, applications, and infrastructure from potential threats. Achieving such a level of security on one's own could require much time and specialized labor.

Such benefits explain why cloud computing has been so widely used nowadays and why this market segment has grown so rapidly.

3.2 How does cloud computing work?

Since cloud computing is not just a technology itself, but a service, its functioning stems mainly from the strategies used to make this service available in a way that the value proposition is convincing and wins enough customers for market maintenance and expansion (ARMBRUST *et al.*, 2009).

As introduced by the previous sections, the key points of the value proposition offered by cloud computing providers are cost reduction, agility, scalability, reliability, and security to have access to the same resources compared to developing them locally (BUYYA *et al.*, 2009).

This means that the resources themselves will be implemented similarly, the main difference is who is responsible for implementing and maintaining these resources.

A detailed explanation of how resources are implemented is beyond the scope of the topic, since the implementation itself does not distinguish much from a local implementation, except for the access control and resource management layer that is especially necessary for cloud computing.

But, in a simplified way, a cloud service provider invests in massive infrastructure, composed of servers, *data centers*, and state-of-the-art network elements. Due to the high investment, these providers can access more advanced, more powerful technologies at a significantly lower cost than would be possible for an ordinary user.

In addition, due to the large scale of infrastructure, cloud computing providers maintain highly specialized teams in each segment (ZHANG; CHENG; BOUTABA, 2010), allowing more efficient configuration of resources than the default configurations often used by local hosting.

Also, these highly specialized teams can offer a more consistent service (less susceptible to errors) and with shorter incident response time.

Thus, from infrastructure with capacity to serve large demands and efficient and reliable configuration, cloud computing providers develop enhanced solutions (PaaS, SaaS, and *serverless computing*) with their own resources and make access to them available through the network (MELL; GRANCE, 2011).

Specialized teams in each area (artificial intelligence, data analysis, development, among others) are responsible for such enhanced solutions, achieving results that in many cases would not be possible for smaller companies.

Finally, cloud computing providers have a supervision, monitoring, access control, and resource management and distribution layer that controls how, when, and in what quantity each customer will have access to each resource.

3.3 What is cloud computing used for?

The flexibility of cloud computing has enabled the development of many distinct solutions, whether through IaaS, PaaS, SaaS, or *serverless computing*. Furthermore, through the types of cloud (public, private, and hybrid), customers can still choose according to their business needs, even considering regulatory issues that may be of interest.

Among the distinct solutions that are available through cloud computing, the following can be mentioned (AMAZON, 2025c):

Computing capacity: is associated with the *hardware* chosen as base infrastructure, including the processor, memory, network interface, and other important configurations. One of the solutions that cloud computing offers is the option to contract the most appropriate hardware according to the user's needs for capacity/processing power, being able to opt for infrastructure with a more powerful processor and more processing cores (for example), or for infrastructure whose RAM memory is larger, or even a combination of both.

Storage: is another solution that cloud computing providers offer, through different database options, whether structured (*Structured Query Language* (SQL)) or unstructured (NoSQL), in addition to *backup* and maintenance. Through this solution, users can choose from a series of available databases, as well as define their storage capacities, and update their sizing as needed.

Artificial intelligence and machine learning: are also part of the scope of solutions offered by providers, in this case offering access to ready-made artificial intelligence solutions, such as code generation assistants, autonomous agents, *chatbots*, natural language processing solutions (*Natural Language Processing* (NLP)) and audio-to-text conversion (and vice versa).

Network solutions: allow the availability and delivery of applications, files, and data to anywhere in the world with high levels of availability, in a distributed manner, with the ability to automatically adapt to demand (network traffic) and through very high speeds. A user, for example, who wanted to make a file available from a local server, could face availability problems (if the server became unavailable), so that files would become inaccessible, could face latency problems when making the file available to different regions of the world, could face problems with demand, if a high number of downloads of the same file were requested simultaneously, and could even face problems with speed, if the internet connection was not a high-speed connection. All these problems are reduced or even solved through network solutions from cloud computing providers.

These are just some of the many solutions that cloud computing providers offer. But, among these solutions, there is a segment that is of special interest for the development of this work, which are cloud computing solutions for computer vision.

3.4 Cloud computing solutions for computer vision

Cloud computing providers have developed specific solutions for computer vision. Different providers offer slightly different solutions, so an evaluation at the provider level is needed to know the possibilities offered by each one and how they compare.

3.4.1 AWS Amazon Rekognition

The cloud computing services provider Amazon has a range of solutions related to computer vision as part of *Rekognition*.

Amazon *Rekognition* is an image and video analysis service that makes it easy to add advanced computer vision capabilities to applications, without requiring the user to have any prior experience with machine learning.

Among the functionalities offered by Amazon *Rekognition* can be mentioned:

- Detection of objects, scenes, and contexts in images or videos;
- Detection and extraction of text from images or videos;
- Detection and filtering of unsafe content in images or videos;

- Recognition of celebrities in images or videos;
- Facial analysis and identification of attributes like gender, age, and emotions in images or videos;
- Creation of custom models to identify specific objects in images;
- Analysis of image attributes (quality, color, sharpness, contrast, etc);
- Analysis of people's trajectory in videos;
- Video segmentation.

As observed from the list above, there is the possibility of accessing specific functionalities for applications with images or for applications with videos. These functionalities are accessed through different APIs[3].

APIs can be synchronous or asynchronous. In synchronous APIs, the API receives an input, processes this input, and returns a response. It is not possible to advance to the next step of the application while the response is not returned. This behavior causes momentary blocking in execution, but does not harm the operation, since it is normally used for APIs that provide fast responses. On the other hand, in asynchronous APIs, the API receives an input (or a batch), and performs processing of these in the background, allowing the application to advance to the next steps. When the API finishes processing, it notifies the application that the response is ready to be returned, thus reducing blocking times. This alternative is, in general, more used for APIs whose response is slower.

For Amazon *Rekognition* image API functions synchronously, that is, an image is sent, analyzed, and a response is returned.

Another characteristic of APIs is the storage or not of data related to the received input. Most image and video analysis APIs do not store data related to the input, but some do this to offer more advanced functionalities. For example, a face detection-based authentication application requires registration of faces to which access will be allowed, in which case storage of data related to inputs becomes necessary.

Images provided to the API must be in .jpg or .png format. And must be made available either through a reference to Amazon S3 (file storage solution) or through *bytes* of images loaded from the local system.

Similarly, there is an API for video analysis. A substantial difference is that the video needs to be stored in Amazon S3, which means that a preliminary step to send the video to

[3] An *Application Programming Interface* (API) is a set of rules and definitions that allows different software systems to communicate with each other. It defines methods and protocols for software to request and send data to another, abstracting the complexity of internal implementation. In a simplified way, an API is a path (*link*) for access to a service defined according to a set of rules.

Amazon S3 is necessary. The file format must be MOV or MPEG-4, with H.264 encoding and the maximum size must be 10 GB.

Another substantial difference about the video API is that its operation is asynchronous, since video analysis requires more processing time.

These are the fundamental characteristics of the APIs made available by Amazon Rekognition, through these APIs some operations can be performed from object detection in images to analysis of people's trajectory in videos (Amazon Web Services (AWS), 2025b).

3.4.1.1 *Object and concept detection in images or videos on AWS*

A label is used to identify an object or concept in an image. A concept also includes scenes/environment and actions. For example, an image of people on the beach may contain labels such as: palm tree (object), beach (scene), running (action), and outdoors (concept).

The current available version can identify up to 3,083 different objects or concepts. Labels are by default in English, which can be translated to any language with some additional steps. But, object localization in the image is restricted to 288 objects. This approach makes sense since actions, scenes, and concepts cannot be located in an image using a rectangular box and represent an understanding or comprehension of the whole. This functioning can be observed in Figure 9, where an analyzed image presents several objects located and identified by rectangular boxes, and the label of *Person* for the person.

Figure 9 – Image analyzed by Amazon *Rekognition* object detection service.

Source: Amazon (2025c).

Among the labels, there is a hierarchical relationship of parentage, for example, a car is a vehicle, but a motorcycle is also a vehicle, as is a truck. An analyzed image that has a car, a

motorcycle, and a truck, will return respectively, vehicle>car, vehicle>motorcycle, vehicle>truck. In this representation, the item that precedes the ">" character identifies the parent element and the item that follows is the child. Still, labels are grouped according to categories and subcategories, such as "Clothing" and "Food".

Despite the large number of labels (objects and concepts), only 3,083 are not sufficient to describe many specific applications. For example, among the 3,083 pre-existing labels there are some dog breeds, such as Chihuahua, Bulldog, and Pitbull, but there are no Dobermann, Shih-Tzu, or mixed-breed dogs, for example. An application to detect and classify dogs according to their breeds would be limited if it relied only on pre-existing resources.

To work around this problem, Amazon offers *Custom Labels* (Amazon Web Services (AWS), 2025a) or custom labels. An auxiliary service that allows the user, from a small set of training images (on the order of hundreds of images), to create a custom detection model. This service, however, uses a specific API, and has special pricing proportional to training time and model usage time.

3.4.1.2 Amazon Rekognition pricing

Service prices vary according to the solution used, according to solution configurations, according to processing region, and according to the number of API calls.

For Table 2 the US East (Ohio) region was considered.

In addition to the costs above, which are the costs involved in image analysis, APIs that result in metadata storage, such as those that allow face search, incur additional costs related to storing this metadata.

The metadata storage cost is $0.00001 per face/month and applies to the storage of face and user vectors. For example, suppose that for user A, there are stored face metadata for 3 faces. Such a scenario represents 4 cost instances, being 1 for the user vector and 3 for the face vectors. Billing is monthly, so only a proportional amount will be charged for the time used in that month.

Table 2 also demonstrates behavior in which the analysis price per image decreases as a function of the number of images analyzed in a one-month period, benefiting customers who analyze large quantities of images.

Video analysis pricing also depends on the type of analysis, since *Rekognition* supports analysis of videos stored in Amazon S3 and also video *streaming*.

Although the pure price for label detection in video *streaming* is slightly lower, there are hidden associated costs that contribute to increasing the value. Among these associated costs is the cost for data ingestion into the *stream* ($0.0085/GB), the cost for data consumption from the *stream* ($0.0085/GB), and the cost for *stream* storage ($0.0085/GB-month).

Stored video analysis also has additional associated costs, such as the cost for Amazon

Table 2 – Amazon *Rekognition* prices by cost category in USD for each image analysis request by accumulated quantity of image analysis requests performed in the month.

Cost Category	Category Solutions	0 to 1 million requests	1 to 5 million requests	5 to 35 million requests	above 35 million requests
1	AssociateFaces, CompareFaces, DisassociateFaces, IndexFaces, SearchFacesbyImage, SearchFaces, SearchUsersByImage, SearchUsers,	$0.001	$0.0008	$0.0006	$0.0004
2	DetectFaces DetectModerationLabels* DetectLabels** DetectText RecognizeCelebrities DetectPPE	$0.001	$0.0008	$0.0006	$0.0004
3	Image Properties***	$0.00075	$0.0006	$0.00045	$0.0001875

Source: Amazon (2025a).

Note *: Allows bulk image analysis, but each image is considered a cost instance

Note **: The *DetectLabels* API offers options to specify some configurations, this price refers to when the GENERAL_LABEL configuration is triggered.

Note ***: Another option that the *DetectLabels* API offers is IMAGE_PROPERTIES, which when triggered incurs this price.

S3 storage. Amazon S3 prices are also highly variable, starting from $0.0023 in the standard option.

From Table 3 it can also be inferred that many solutions or functionalities are not available in stream analysis.

The video shot detection and technical cue detection functionalities are priced independently, but are called through the same API, allowing the user to specify whether they want to perform both detections or just one of them.

Other functionalities can also be combined, but from calls to different APIs for the same video. For example, calling the API for label detection and then the API for text detection.

In addition to image and video analysis functionalities, Amazon *Rekognition* also offers the functionality to train a custom model for object detection. This option has its own pricing model that is divided into training hours and inference hours, as shown in Table 4.

In this table, it is observed that the inference cost is per hour and per *compute*, which indicates the possibility of having multiple active processing instances of the model to accelerate

Table 3 – Amazon *Rekognition* prices by solution available in the API for each implementation architecture (*streaming* and stored video) in USD for video analysis.

Solution	*Streaming*	Stored Video
Label Detection	$0.00817/min	$0.10/min
Content Moderation	NA	$0.10/min
Text Detection	NA	$0.10/min
Face Detection	NA	$0.10/min
Celebrity Recognition	NA	$0.10/min
Face Search	NA	$0.10/min
Video Shot Detection	NA	$0.05/min
Technical Cue Detection	NA	$0.05/min

Source: Amazon (2025a).

Note: Fields indicated by *not applicable* (NA) in the *streaming* column indicate that the corresponding solution is not available for streaming architecture.

image processing.

Also, the inference cost is proportional to the number of hours the model is active to be used, and not necessarily to the number of hours the model was used. For example, a model that was active for 1 hour, but processed no images, will still have the cost of $4.00. Therefore, it is recommended to keep models deactivated, unless there are images to be processed.

Table 4 – Amazon *Rekognition* prices in USD for model customization by service.

Service	Price
Inference	$ 4.00/hour-*compute*
Training	$ 1.00/hour

Source: Amazon (2025a).

Some cloud computing providers offer some free functionalities in a limited way. For *Rekognition* there are also some of these, such as:

- free image analysis up to the limit of 1,000 images per month, for a period of one year;
- free face and user vector storage for up to 1,000 vectors of each, per month, for a period of one year;
- free video analysis up to the limit of 60 minutes of video per month, for a period of one year;
- free model customization up to the limit of 2 hours of training and 1 hour of inference per month, for a period of one year.

3.4.2 Azure AI Vision

Microsoft also has its own segment of solutions related to computer vision, these solutions are part of *Azure AI Vision*, a service that provides access to advanced image processing algorithms distributed among 4 categories: OCR, image analysis, facial recognition, and video analysis.

A characteristic in which *Azure AI Vision* distinguishes itself from Amazon *Rekognition* is in the size and variety of supported input files.

While the *Rekognition* API allows only .png or .jpg files, the *Azure* API also accepts files in .gif, .bmp, .webp, .ico, .tiff, and .mpo formats. This flexibility can represent a competitive advantage over *Rekognition*.

Also, *Azure* accepts images up to 20 MB in version 4.0 of its API, while *Rekognition* can work with images up to 15 MB, when provided through Amazon S3, or up to 5 MB, when provided in *bytes* format, this is another difference that can represent a potential limitation of *Rekognition* compared to *Azure*.

Regarding the API operation mode, APIs for image analysis are synchronous and APIs for video analysis are asynchronous, just like in *Rekognition*.

Among the services that comprise *Azure AI Vision*, OCR is the alternative for text recognition and extraction, which can be performed on images or documents, but through different tools.

The image analysis service allows detection of people and objects, caption or label generation (identify image context), brand detection, image categorization, face detection, adult content moderation, among other functionalities.

The face detection service is also quite similar to what is offered by Amazon, comprising options for user identity validation, liveness detection, access control, among others.

A particularity of *Azure* face detection service compared to *Rekognition* are policies that limit service functionality according to data protection laws and according to possible risks associated with technology misuse. For example, functionalities for emotion detection and binary gender classification were removed, and functionalities for age detection, smile, facial hair, hair, and makeup were limited. These functionalities can still be requested via email to Microsoft, but are not available to users in general (AZURE, 2022).

The decision to limit and remove these functionalities aims to promote more responsible use of artificial intelligence solutions, but represents a limitation of the *Azure* tool compared to *Rekognition*, which still offers among its solutions, for example, binary sex classification and emotion classification.

The video analysis service was removed from *Azure AI Vision* to be made available from its own solution, *Azure AI Video Indexer*.

Azure AI Video Indexer uses artificial intelligence to analyze videos, offering functionalities such as object detection, face recognition, audio transcription, emotion analysis, and much more.

Unlike competitors (Amazon and Google), *Azure AI Video Indexer* provides a solution for multimodal analysis (video, audio, and text), while competitors provide natively only video analysis, so to analyze other types of input, calls to auxiliary APIs for processing and analysis of this content are necessary.

3.4.2.1 Object and concept detection in images or videos on Azure

Unlike Amazon *Rekognition* whose documentation specifies the number of objects or concepts that can be detected in an image, this information is not available in *Azure* detection[4], making direct comparison between the two cloud computing providers impossible in this aspect.

Despite this, the category list is available in *Azure* with a total of 86 categories, which exceed the 40 categories available in *Rekognition*.

The same functionality of detecting actions, scenes, and concepts, in addition to objects, is also present. As well as image categorization.

Another functionality that *Azure* offers in common with *Rekognition* is the custom model training tool, to allow detection of specific objects that are not among those that the original image analysis API model can identify.

However, just as the video analysis service was removed from *Azure AI Vision* to be its own solution, *Azure AI Video Indexer*, the same happened with the solution for creating custom labels, where the tools that enabled this functionality were removed from *Azure AI Vision* and now comprise a new Azure solution, *Azure AI Custom Vision*.

The *Azure AI Custom Vision* service uses a machine learning algorithm to analyze images, where the user sends sets of images that have and do not have the visual characteristics being sought. Then, the images are labeled by the user. Following this, the algorithm trains a model for this data and calculates precision by testing on the same images. Models can be retrained as needed for improvements.

An important difference compared to the service offered for *Rekognition*[5] is the possibility of exporting the model for offline use, which enables use for edge computing.

[4] *Azure* documentation only informs that there are "thousands" of labels, without specifying the quantity.

[5] Although *Rekognition* does not offer the possibility to export its models, this, however, does not mean that it is impossible to export models from AWS. What happens is that this functionality is not part of *Rekognition* scope and is available only in other solutions such as *Amazon SageMaker*.

3.4.2.2 Azure pricing for Computer Vision

Just like for Amazon *Rekognition*, prices vary according to solution, solution configuration, number of API calls, and processing region.

According to Table 5, Microsoft also offers discounts based on increased number of images to be analyzed.

Table 5 – *Azure AI Vision* prices by cost category in USD for each image analysis request by accumulated quantity of image analysis requests performed in the month.

Cost Category	Category Solutions	0 to 1 million requests	1 to 10 million requests	10 to 100 million requests	above 100 million requests
1	Tag, Face, GetThumbnail, Color, ImageType, GetAreaOfInterest, SmartCrops, OCR, Adult, Celebrity, Landmark, ObjectDetection, Brand	$0.001	$0.00065	$0.00060	$0.0004
2	Describe Read Caption DenseCaptions RecognizeCelebrities DetectPPE	$0.0015	$0.0006	$0.0006	$0.0006

Source: Azure (2025a).

The video analysis API offers 3 options: pure audio analysis (without video), pure video analysis (without audio), and audio and video analysis.

For both, there are 3 available categories: basic, standard, and advanced, according to Table 6.

The training and inference service with custom models is also available in Azure, through *Azure AI Custom Vision*, according to Table 7. In this service there is an additional cost related to images stored for model training which is $0.0007 per image.

Like Amazon, Azure also offers some free functionalities in a limited way, for example:

- free image analysis up to the limit of 5,000 images per month or a maximum of 20 transactions per minute;

Table 6 – *Azure AI Video Indexer* prices by media type and analysis category (USD/min).

Media Type	**Category**	**Included Features**	**Price (USD/min)**
Audio	Basic	Audio transcription, translation, caption and subtitle generation	$ 0.0126
Audio	Standard	Language detection, emotion detection, keyword detection, topic extraction, content moderation	$ 0.024
Audio	Advanced	Event detection	$ 0.04
Video	Basic	Object detection, label definition, text extraction, black frame detection	$ 0.045
Video	Standard	Labels and entities from text extraction, content moderation, people detection	$ 0.09
Video	Advanced	People identification, digital pattern detection, clothing detection, logo detection	$ 0.15
Multimodal	Basic	Combination of audio and video analysis in Basic category	$ 0.0576
Multimodal	Standard	Combination of audio and video analysis in Standard category	$ 0.114
Multimodal	Advanced	Combination of audio and video analysis in Advanced category	$ 0.19

Source: Azure (2025b).

Table 7 – *Azure AI Custom Vision* prices in USD for model customization by service.

Service	Price
Inference	$ 0.002/image
Training	$ 10.00/hour

Source: Microsoft Azure (2025).

- free video analysis up to the limit of 40 hours of video indexing through the API and 10 hours through the website;

- free model customization up to the limit of 1 hour of training and 10,000 predictions (inferences).

3.4.3 Google Cloud Vision

Another cloud computing provider that has its own segment of computer vision solutions is Google, which through Google *Cloud Vision* and other auxiliary services, provides users with various image and video analysis tools.

Among the functionalities offered by Google *Cloud Vision* can be mentioned: detection and extraction of texts (OCR) in images and documents, detection of objects and scenes, including logos, landscapes, faces, and others, extraction of image properties, and content moderation.

Like *Azure*, *Cloud Vision* is capable of working with a great variability of input file formats, such as: .jpg, .png, .gif, .bmp, .webp, .raw, .ico, .pdf, and .tiff.

This variability matches Azure and offers more options than those available in *Rekognition*.

On the other hand, while both Azure and *Rekognition* require images with minimum size of 50 x 50 pixels, Google *Cloud Vision* requires images of at least 640 x 480 pixels. This requirement can make using the tool unfeasible for lower resolution images.

Regarding the size in *bytes* supported for images, the limit is 20 MB. However, this limit can only be reached if images are stored in *Google Cloud Storage*. For use through the API the limit is 10 MB.

For *Cloud Vision*, also similarly to *Azure* and *Rekognition*, there are 2 different APIs, one for image analysis and another for video analysis.

Also similarly to competitors, the API for video analysis works asynchronously, given the nature of the request (videos are heavier, and therefore, require longer processing time, so that synchronous API calls could be limiting). But, for the image analysis API there is a difference. While the *Azure* analysis API is essentially synchronous, in *Cloud Vision* the API has both options, synchronous and asynchronous, where synchronous is used for analysis requests with 1 image and asynchronous is used for batch analysis requests (multiple images). *Rekognition* also has the batch image analysis function.

Among the functionalities that *Cloud Vision* offers, can be mentioned:

- Detection and extraction of texts using OCR in images and documents;
- Detection of crop hints;
- Face detection;
- Detection of Image Properties (captures image characteristics regarding colors);
- Detection of Logos, Landmarks, and Objects;
- Content Moderation;

- Object Detection.

In face detection, it is possible to identify multiple faces in an image and returns attributes such as emotional state and use of accessories (glasses, hat, etc).

The functionality to detect emotions is also available in *Rekognition*, but is not available in *Azure*. And the functionality to detect individual faces (associate with a user, for example), is not available in *Cloud Vision*, but is in competitors.

Google's video analysis solution is offered through *Vertex AI Vision* (similar to Microsoft that separated *Azure AI Vision* from *Azure AI Video Indexer*).

Vertex AI Vision is a platform that allows ingesting, analyzing, and storing video data, including functionalities for object detection, people and vehicle tracking, motion filtering, among others.

Videos for analysis can be provided through *streaming* or can be stored in Google *Cloud Storage* and analyzed from there.

3.4.3.1 Object and concept detection in images or videos on Google Cloud

In *Cloud Vision*, there are two different APIs, one for object localization in the image and another for image labeling. Object localization is capable of detecting and identifying in the image the location of physical objects belonging to about 100 known classes, such as (person, car, bicycle, dog, cat, etc). Like Amazon and Google, Google can also recognize concepts, which cannot be located at a specific position in the image. For this, the second API is used, specific for image labeling.

Considering the objective - detection of people and vehicles - the object localization API is most appropriate.

Cloud Vision documentation does not precisely inform the number of distinct labels/objects that can be detected. But, according to Google itself, this model can detect more than 550 different labels (Google Research, 2025).

Generally, *Cloud Vision* documentation is quite lean compared to *Rekognition* and *Azure* documentation, but one of the available pieces of information is about the model architecture, which uses a *single-shot* type detector (detection and classification in one step, similar to YOLO) (PARAB *et al.*, 2022), with resnet-101[6] and a feature map based on a Feature Pyramid Network (FPN)[7].

[6] ResNet-101 is a deep neural network used for image feature extraction, where the numbering 101 indicates the number of depth layers of the network (HE *et al.*, 2015).

[7] It is a technique that improves object detection at different scales (small, medium, and large), through creation of multiple feature maps with different levels of detail, allowing highlighting from small objects to large objects (LIN *et al.*, 2017).

Google also offers its own solution for training custom models. This solution can be implemented in two ways: 1) through *AutoML Vision*, a no-code tool[8] where the user uploads images, labels objects, and then trains the model; or 2) through *Vertex AI Custom Training*, which allows creating completely customized models using TensorFlow, PyTorch, or Scikit-learn, adjusting hyperparameters and with full control over model architecture.

3.4.3.2 Google Cloud pricing for Computer Vision

In terms of pricing, the *Cloud Vision* pricing model is similar to competitors, with lower prices based on increased usage demand.

Prices for image analysis are described by Table 8.

Table 8 – Google *Cloud Vision* prices by cost category in USD for each image analysis request by accumulated quantity of image analysis requests performed in the month.

Solution	0 to 1 thousand requests	1 thousand to 5 million requests	above 5 million requests
Label Detection	Free	$ 0.0015	$ 0.001
Text Detection	Free	$ 0.0015	$ 0.0006
Content Moderation	Free	$ 0.0015	$ 0.0006
Face Detection	Free	$ 0.0015	$ 0.0006
Image Properties	Free	$ 0.0015	$ 0.0006
Object Detection	Free	$ 0.00225	$ 0.0015

Source: Cloud (2025).

For video *stream* analysis, prices are defined according to Table 9. Google offers a pricing model not practiced by direct competitors, with the option of monthly billing per *stream*, instead of only consumption billing (minute analyzed), as occurs with Amazon Rekognition Video and Azure Video Indexer.

For *streaming*, additional costs of $0.0085/GB for ingestion and $0.0085/GB for video consumption occur, regardless of whether the user opts for consumption pricing or monthly pricing.

Alternatively, if the user decides to perform processing from videos hosted on Google, at a price of $0.020/GB-month, whether for consumption pricing or monthly pricing.

Also, prices for customization or model personalization are similar in structure to competitors, according to Table 10. It should be noted, however, that the inference price of $0.22/hour is the lowest available price. There are options up to $101.00 for servers with more CPU, GPU, and memory resources.

[8] No-code tools allow creating applications, automations, and even AI models without needing to program and without advanced technical knowledge.

Table 9 – Google *Vertex AI Vision* prices in USD by service offered for video analysis for each pricing category (pay-as-you-go and monthly).

Service	Pay-as-you-go Price	Monthly Price
People and Vehicle Counting	$ 0.10/min	$ 10/stream
Object Detection	$ 0.10/min	$ 10/stream
PPE Detection	$ 0.10/min	$ 10/stream

Source: Cloud (2025).

Table 10 – Google *Vertex AI Vision* prices in USD for model customization by service.

Service	Price
Inference	$ 0.22/hour
Training	$ 2.00/hour-*compute*

Source: Cloud (2025).

3.5 Cloud computing solutions for running custom applications

Although cloud computing providers provide a range of ready-made solutions in the field of computer vision, such as object detection, text extraction, and other functionalities already mentioned, these ready-made solutions do not cover all existing scenarios.

For these and other reasons, cloud computing providers also offer a series of solutions that allow the development of customized applications according to the needs of a specific problem.

Among these solutions, two main approaches stand out: direct allocation of virtualized infrastructure and managed container execution services.

The first approach is represented by solutions such as Amazon EC2, Azure Virtual Machines, and Google Compute Engine, which provide virtual machines with full control over the operating system, network resources, storage, and scalability. This option offers maximum flexibility and is suitable for highly customized processing workloads, but demands greater responsibility for configuration, security, load balancing, scaling, and infrastructure maintenance, which generally requires more knowledge and time from the person responsible for environment configuration.

On the other hand, providers also make available managed services for container execution, aimed at applications that require less operational intervention and greater elasticity. In this context, *Azure Container Apps*, *AWS App Runner*, and *Google Cloud Run* stand out. These services offer an abstraction layer over infrastructure, with automatic resource provisioning, dynamic scaling based on demand, billing based on actual CPU and memory consumption, and integration with other native cloud services.

The choice between these approaches depends directly on application characteristics, necessary control over the execution environment, and scalability and cost requirements. This work focuses on evaluating and comparing managed *container* solutions offered by the three main cloud platforms (Azure, AWS, and Google Cloud), with the objective of understanding cost and performance behavior in executing custom computer vision applications.

3.5.1 AWS App Runner

AWS App Runner is a managed service that enables rapid and simplified deployment of applications from source code or *container* images. Its main objective is to eliminate infrastructure provisioning complexity, allowing developers and operations teams to focus on development and software delivery, without the need to configure servers, load balancers, or manual *Continuous Integration/Continuous Deployment* (CI/CD) pipelines (AMAZON, 2025b).

App Runner has functionalities that allow direct connection with code repositories like GitHub or image repositories like Amazon *Elastic Container Registry* (ECR), automatically executing *container* construction and continuous delivery of new versions with each repository update, if this option is enabled.

In terms of scalability, App Runner automatically adjusts the number of active *container* instances according to the volume of received requests. It is possible to define minimum and maximum limits of provisioned and active instances, which provides control over performance and cost. When there is no traffic, the service maintains one or more provisioned instances (with reduced cost) to avoid initialization latency, eliminating the "cold start" problem.

App Runner's billing model is based on two operational states:

- Provisioned instances: when the application is idle, only allocated memory is charged (for example, $0.007 per GB-hour). These instances remain "warm" to ensure rapid response to requests.

- Active instances: when the application is processing requests, CPU usage ($0.064 per vCPU-hour) and additional memory consumed are also charged. Billing is per second, with a minimum of one minute per activation event.

3.5.2 Azure Container Apps

Azure Container Apps is Microsoft's solution that allows running applications in *containers* without the need to manage servers (AZURE, 2025).

Like AWS App Runner, Azure's service abstracts infrastructure complexity, allowing developers to focus on the application.

Azure Container Apps offers a consumption plan where computational resources (vCPU and memory) are charged based on actual usage per second.

Billing is divided into two states:

- Active usage: when a replica is processing requests, standard billing is $0.000024 per second per vCPU and $0.000003 per second per GiB of memory.
- Idle usage: when the replica is active, but without significant load, the cost is reduced ($0.000003 per second per vCPU and per GiB of memory).

The service allows configuring minimum replicas (keeping the application "warm") or scaling to zero when there is no demand, eliminating costs during periods of total inactivity. This makes it especially attractive for intermittent or unpredictable workloads.

3.5.3 Google Cloud Run

Google Cloud Run is another managed platform that allows running *container* applications without the need to configure *clusters*, balancers, or servers (GCP, 2025).

In the context of *Hypertext Transfer Protocol* (HTTP) request-based applications, Cloud Run is particularly attractive, as it allows automatic scaling based on requests, where the service scales horizontally the number of instances according to request volume, and can even scale to zero when there are no requests, reducing cost during periods of inactivity.

Cloud Run adopts a granular usage-based billing model based on computational resource usage time (vCPU and memory) and number of requests. The main values are $0.00002400 per second of vCPU and $0.00000250 per GiB-second.

Billing occurs only during the time instances are actively processing requests. Usage is calculated with 100-millisecond precision, and there is no charge when the service is inactive (if no minimum instance is configured).

CHAPTER

4

COMPUTAÇÃO DE BORDA E EMBARCADA

In the previous chapter, the main concepts about cloud computing were highlighted, the main cloud computing providers in the market and what solutions each of them offers regarding computer vision.

But, despite being very important, cloud computing is not the solution to all problems, so there are many circumstances in which cloud computing may not be the ideal solution. In fact, cloud computing is still a recent technology, so some solutions that are still used already existed even before the popularization of cloud computing. These solutions have also evolved and reinvented themselves over the years to remain attractive to the market.

Among these alternatives to cloud computing that already existed or that evolved to meet requirements that cloud computing could not, one can cite embedded computing and edge computing.

Embedded computing refers to computational systems integrated into hardware used for real-time control and processing, optimized to meet specific application requirements, such as power consumption, limited size and lower cost (OSHANA, 2006). An important characteristic of embedded systems is that they are reactive, in the sense that embedded systems act as a response to stimuli received through sensors, for example. Some examples of embedded systems are: vehicle engine controllers, intelligent sensors and medical equipment.

Edge computing is a development pattern in which data processing is kept close to the generation source, without the need to send data to a centralized data center to perform processing. This programming paradigm is especially interesting for applications that require lower latency, lower bandwidth, more efficiency, and fast responses. Some examples are: autonomous vehicles, augmented reality and internet of things (SHI *et al.*, 2016).

These concepts - embedded computing and edge computing - are not opposites, but rather complementary. An embedded system can be designed to send data for decentralized processing,

usually using cloud computing, or can be designed to perform processing locally, usually using edge computing.

This architecture can be observed, for example, in a drone with sensors for image capture, which processes detections of interest in the embedded hardware and sends to the server (cloud) only the post-processing results, or an industrial sensor that monitors a machine in real-time, performing prediction processing of stops in the embedded hardware and sending only alarms to be processed by the server (cloud).

The popularity of solutions that combine embedded computing with edge computing has increased due to multiple factors, among which some can be cited:

- Increased use of internet of things devices: the greater number of devices means a proportionally greater number of data to be processed, so sending all this data to the cloud for processing becomes unfeasible due to latency, bandwidth and associated costs (GUBBI *et al.*, 2013);

- Latency in critical applications: in some applications the time between sending data for cloud processing, data processing and sending the response can mean unacceptable latency for solution requirements. For example, higher latency can mean longer response time, and in an autonomous car this difference could cause accidents (SATYANARAYANAN, 2017);

- Advances in specialized hardware: one of the reasons for performing cloud processing was the possibility of accessing more powerful computational resources, so that, even if there was latency for sending the request and receiving the response, cloud processing would be so much faster that it would compensate for the latency and still result in faster processing than that performed at the edge. However, in recent years, the hardware solutions available for edge computing have evolved and offer great processing power that can again enable edge processing (OLIVEIRA *et al.*, 2024).

Due to these and other factors, the use of edge computing has increased, but there are also limiting factors, so edge computing is also not a definitive solution that can meet all types of requirements.

For example, although there has been great evolution in hardware systems used for edge computing and processing power has increased in recent years, the servers available by cloud computing providers are still much more powerful.

Also, there is the impossibility of updating hardware resources. Once the embedded system is designed with certain software and hardware, it will not be possible (or will be very difficult) to update its hardware to a more powerful version without complete device replacement.

A third limitation of using embedded systems with edge computing is the higher initial investment, both in hardware and in qualified workforce for development, assembly and device configuration.

According to such advantages and disadvantages, the decision to use embedded computing architecture with edge computing or the decision to use cloud processing depends greatly on the requirements of the solution to be developed. In many cases, a hybrid approach is used. Combining the advantages of edge computing with the advantages of cloud computing. As an example, one of the areas that greatly benefits from edge computing is the area of computer vision.

4.1 Edge computing in computer vision

Among the many applications that benefit from edge computing is computer vision.

Some of the reasons why the use of edge computing is suitable for computer vision solutions are: lower processing latency that enables fast responses, essential for critical systems, such as autonomous cars; lower internet bandwidth consumption, which when very high makes transmission to the cloud unfeasible, such as in video streaming; independence from network connectivity for processing, such as in spraying drones, which in the absence of connection, could lose the scanning state and not know where to continue spraying.

These examples highlight some requirements that have greater weight in determining whether edge processing is an approach that makes sense or not for a given application. These requirements are: latency, bandwidth, reliability and cost.

4.1.1 Latency requirements that enable edge computing

Although the specific definitions of these requirements vary according to the application, in general, some limit values are used more frequently.

In industrial automation applications, latency is considered low if it is below 10 milliseconds; in traffic control services on roads, latency below 100 ms is acceptable; in professional gaming, latencies must be less than 50 ms; these and other examples can be observed in Table 11.

4.1.2 Bandwidth requirements that enable edge computing

The same applies to bandwidth (the amount of data that can be transferred through the network per second). In computer vision, especially, different cameras produce files with different sizes, proportional to their resolution. For example, a Full HD camera capturing video at 30 frames per second generates approximately 5 Mbps of data. On the other hand, a 4k camera recording at 60 frames per second generates approximately 20 Mbps of data.

Table 11 – Typical latency and data rate requirements for different applications.

Application	Latency	Data Rate	Observations
Industrial Automation	0.25-10 ms	1 Mbps	Factory automation applications generally require small data rates for motion control and remote control. Mechanical tool operations may allow latencies as low as 0.25 ms.
Transportation Systems	10-100 ms	10-700 Mbps	Road safety requires latency in the order of 10 ms. Applications such as virtual mirrors* require data rates in the order of 700 Mbps.
Robotics and Telepresence	1 ms	100 Mbps	Touching an object with the palm of the hand may require latency below 1 ms. Haptic feedback in VR requires data rates of approximately 100 Mbps.
Virtual Reality (VR)	1 ms	1 Gbps	High-resolution 360º VR requires rates in the order of 1 Gbps with 1 ms latency.
Healthcare	1-10 ms	100 Mbps	Telehealth, telesurgery and telerehabilitation may require 1 ms latency with 100 Mbps data rates.
Performance Gaming	1 ms	1 Gbps	Immersive entertainment and interaction with high-quality graphics may require 1 ms latency and 1 Gbps rates for high performance.
Smart Grid	1-20 ms	10-1500 Kbps	Dynamic activation and deactivation in smart electrical grids require 1 ms latency. Wide area monitoring may require data rates of 1500 Kbps.
Education and Culture	5-10 ms	1 Gbps	Multisensory interfaces enabled for tactile Internet may require latency up to 5 ms. High-resolution 360º VR and haptics may require rates up to 1 Gbps.

Source: Parvez *et al.* (2018).

Note *: Virtual mirror: Camera and screen system that replaces a vehicle's physical rearview mirrors with external cameras that capture rear and side views, and these images are displayed in real time on screens inside the car.

Consider now that a set of 100 cameras is used for monitoring a small city and that all data from these 100 cameras needs to be sent for cloud processing. In the best case (Full HD 30 fps camera), it would represent 500 Mbps, and in the worst case (4k 60 fps camera), 2 Gbps.

Some bandwidth examples are also exemplified in Table 11.

4.1.3 Availability requirements that enable edge computing

Another factor that favors edge processing is service reliability, which can be analyzed in terms of availability.

Cloud computing providers determine in their service contracts an *Service Level Agreement* (SLA), which among other responsibilities and obligations, determines the minimum percentage of service availability. The availability SLA of the main cloud computing providers is available in Table 12 (Amazon Web Services (AWS), 2025b; Microsoft Azure, 2025; CLOUD, 2025).

Table 12 – Availability SLA of cloud computer vision services by provider.

Cloud Provider	Availability SLA
AWS Rekognition	99.0%
Azure AI Vision	99.9%
GCP Vision API	99.9%

Source: Elaborated by the author.

At first, these numbers seem quite high, but as Table 13 shows, they can represent hours of unavailability per month.

Table 13 – Correspondence between SLA and downtime for monthly and annual periods.

SLA (%)	Monthly downtime	Annual downtime
99.0%	7h 18min	87h 36min
99.5%	3h 39min	43h 48min
99.9%	43min 49s	8h 45min
99.95%	21min 54s	4h 22min
99.99%	4min 22s	52min 35s
99.999%	26s	5min 15s

Source: Elaborated by the author.

Furthermore, the availability of an application that uses cloud services is not restricted only to the SLA offered by the provider. External factors, such as client network failures, local infrastructure interruptions and power outages, directly affect the real availability of the system. Thus, the effective availability of the application will always be lower than the nominal SLA of the services used.

When multiple services are used in cascade, the total system availability is given by the product of individual availabilities. If a system depends on n services with availabilities $A_1, A_2, ..., A_n$, the total availability A_{total} is:

$$A_{\text{total}} = A_1 \times A_2 \times ... \times A_n = \prod_{i=1}^{n} A_i \tag{4.1}$$

For example, if an application depends on a cloud service with 99.9% SLA and an internet connection with 99.5% availability, the total availability will be:

$$A_{\text{total}} = 0.999 \times 0.995 = 0.994 \tag{4.2}$$

That is, the application would have an effective availability of 99.4%, lower than the cloud provider's SLA.

When designing cloud-based systems, it is essential to consider the real availability of the application, which results from the combination of SLAs of services used, in addition to external factors such as connectivity and local infrastructure.

As a complement to this analysis, edge processing represents an effective strategy to mitigate the impacts of unavailability caused by external dependencies. By moving part of the processing to local devices — close to the data source — the need for constant communication with the cloud is reduced, which decreases exposure to connectivity failures and excessive latency.

4.1.4 Cost requirements that enable edge computing

Also, although the initial costs for developing a system that uses edge computing are higher, since they include the values for acquiring the hardware that will be responsible for processing, in the long term cloud processing costs tend to be higher.

This is due to the cloud pricing model in which billing is per use, however, with increased usage demand, the total values will be proportionally higher.

For example, consider a computer vision application with a camera operating 24 hours a day, transmitting continuous video at 15 FPS for real-time object or event detection. Inference occurs every second, totaling 86,400 inferences per day.

According to AWS pricing(AMAZON, 2025a), the Amazon Rekognition Video service charges approximately $0.10 per minute of analyzed video.

- Minutes per day: 24 h × 60 min = 1,440 min
- Daily cost: 1,440 × $0.10 = $144.00
- Monthly cost (30 days): $4,320.00

Alternatively, an embedded device, such as the NVIDIA Jetson Nano, can be used to perform inference locally.

- Device cost: $129.00 (one-time investment)

- Electrical consumption: 5W × 24h × 30 days = 3.6 kWh
- Monthly energy cost: approximately $1.50
- Hardware amortization over 12 months: $129 ÷ 12 = $10.75
- Total monthly cost: $1.50 + $10.75 = $12.25

As observed from the results in Table 14, for continuous and intensive workloads, such as 24/7 monitoring, edge computing proves to be significantly more economical than using cloud services.

Table 14 – Monthly estimated cost comparison between cloud and edge by architecture.

Architecture	Estimated Monthly Cost
Amazon Rekognition (cloud)	$4,320.00
Jetson Nano (edge)	$12.25

Source: Elaborated by the author.

4.2 Hardware for edge computing

The choice of adequate hardware is fundamental for the success of embedded computer vision applications. There are different categories of devices aimed at edge computing, varying in processing capacity, energy consumption and cost. Among the most relevant manufacturers, NVIDIA stands out with the Jetson line, the Raspberry Foundation with the Raspberry Pi, and Google with the Coral platform.

The NVIDIA Jetson line devices, such as the Jetson Orin Nano and Jetson AGX Orin models, are aimed at applications that demand high computational power, especially in deep neural network inference. On the other hand, devices like the Raspberry Pi 4 Model B offer a low-cost solution for less demanding applications. The Google Coral Dev Board represents an intermediate option with specific acceleration for inference through TPU (Tensor Processing Unit).

Table 15 presents a comparison between the main devices used in edge computing applications.

The TOPS (trillions of operations per second) metric is commonly used to quantify the inference capacity of devices in deep learning workloads. The device choice should consider not only raw performance, but also factors such as energy restrictions, cost and integration requirements with sensors and cameras.

Table 15 – Comparison of hardware characteristics, price and power of embedded devices used for edge computing.

Device	**CPU**	**GPU/TPU**	**Memory**	**TOPS**	**Price (USD)**	**Power (W)**
Jetson AGX Orin 64GB	12-core ARM v8.2	2048-core Ampere + 64 Tensor Cores	64 GB LPDDR5	Up to 275	1799	15–60
Jetson Orin Nano 8GB	6-core ARM Cortex-A78AE	1024-core Ampere	8 GB LPDDR5	Up to 67	249	7–25
Raspberry Pi 4 Model B	Quad-core Cortex-A72 1.5 GHz	Broadcom VideoCore VI (no DNN acceleration)	8 GB LPDDR4	<1	75	7
Google Coral Dev Board	Quad-core Cortex-A53 + Cortex-M4F	Edge TPU (4 TOPS)	4 GB LPDDR4	4	129	4

Source: Elaborated by the author.

4.3 Computer vision models for embedded devices

Typically, computer vision models, such as Convolutional Neural Networks (CNNs), are computationally intensive, requiring optimizations to run efficiently on embedded devices with limited resources. These optimizations aim to reduce energy consumption, memory usage and inference time, without significantly compromising accuracy.

This is a necessary trade-off to enable the development of applications that will be executed on hardware with limited processing power and, generally, inferior to what would be possible to obtain through servers.

The implementation of traditional convolutional neural networks (CNNs), such as ResNet or VGG, on embedded devices is unfeasible due to their high computational resource consumption. To overcome this limitation, lighter architectures were developed that balance precision and computational efficiency, with YOLO being the main one. Table 16 presents a comparison of the YOLOv11 architecture (latest version) on devices such as Jetson Orin Nano and Raspberry Pi 4.

Table 16 – Performance metrics comparison of YOLOv11 model on different embedded devices (Raspberry Pi and Jetson Orin Nano).

Metric	**Device 1 Raspberry Pi 4**	**Device 2 Jetson Orin Nano**
mAP50-95	$\sim$ 0.60	$\sim$0.60
Inference time OpenVINO	$\sim$ 405.2 ms	$\sim$57.4 ms
Inference time PyTorch	$\sim$ 81.2 ms	$\sim$21.3 ms
Inference time TensorRT (INT8)	-	$\sim$3.9 ms

Source: Ultralytics (2024b).

As observed in the table, the embedded device for edge computing use has great influence on inference time, that is, how fast the image can be processed. But, there is little influence on the precision or quality of the model.

CHAPTER

5

METODOLOGIA

This work began with a bibliographical review of the 3 central research topics: computer vision, cloud computing, and edge computing.

From this foundation, the example problem was outlined, which consists of developing a solution capable of performing video analysis and providing as results data on people and vehicle counting. As context for this solution, the following scenario can be exemplified: in the real estate sector, companies that work with retail and customer service repeatedly seek to strategically position the location of their physical stores in order to achieve greater visibility and, as a consequence, more customers. In general, commercial real estate brokers have certain empirical knowledge through their associates, but there are few standardized and statistical means to commercially evaluate the viability of a commercial point. This scenario makes comparison between options difficult for contracting companies and makes pricing difficult for real estate agents.

An application capable of correlating key data and composing evaluation metrics presents a possible solution to this problem. In order to systematize evaluations and allow comparability between results.

Another alternative to this problem already used as a means of balancing decisions in some retail chains is field research, in which an agent representing the company travels to points of interest for leasing at predetermined times and manually performs the counting of people and vehicles that pass through the location, repeating this activity for each point of interest during the same time interval.

Despite being functional, this alternative presents some challenges, such as the availability of an agent to perform such counting, the operational costs involved in this process, the limitation of collection to short and specific periods, susceptibility to human errors during manual counting, in addition to the difficulty in replicating this process in a scalable way and in multiple locations.

Given this context, the use of solutions based on computer vision emerges as a viable

and promising alternative, since it allows automation of collection and analysis of this data in a continuous, precise, and scalable way. However, this approach also imposes its own technical challenges, especially related to video processing, the high volume of data generated, and the need to balance the operational costs of this infrastructure.

In this sense, the comparison between the use of edge computing and cloud computing becomes valid in order to enable a choice that reduces operational cost in the circumstances addressed.

This example problem thus defines an applied and experimental research approach, in which the applied research components come from the solution to be developed, while the experimental research components result from the comparison of different solutions, testing and analysis of hypotheses.

In this way, the theme is developed through a quantitative approach, aiming to eliminate author bias and provide foundation for future research.

Although the described scenario is from the real estate sector, the developed solution is generic and applicable to various contexts that demand image analysis for detection and counting.

The main hypothesis is that edge processing presents lower latency and reduced operational costs compared to cloud processing, but with performance loss in accuracy metrics.

5.1 Application architecture

The proposed application to solve the problem is composed of:

Backend module developed in Python that characterizes an image processing service that receives through an API the video file to be processed. This video file is then distributed to 4 different analysis segments: 1) processing via Azure, 2) processing via AWS, 3) processing via GCP, and 4) edge processing. The results are then treated so they can be compared with each other. For each of the analysis segments, some key metrics are annotated that are used as part of the solution evaluation: 1) response latency for image analysis, 2) time for complete video processing, 3) number of frames processed per second (fps), 4) counting of people and vehicles, 5) absolute percentage error, 6) analysis cost. The evaluation metrics are saved in a database instance. The final report is returned as a composition of key-values to be displayed to the user through a front-end.

Frontend module developed in React-native that characterizes the application interface through which the user sends videos for processing and receives the results.

The Python version used in application development was 3.11. Other auxiliary technologies were used in development, such as Go as backend for the video file upload page for processing, Postgres as database, and docker for simulation of virtual environments.

The use of docker allowed simulation of controlled and reproducible environments. The Go backend was exclusively responsible for video upload. Postgres stored the metrics captured during experiments.

5.2 Implementation of object detection and tracking tasks

For each analysis segment, 2 tasks are performed with the objective of measuring different metrics. Task A represents object detection in each video frame and task B represents object tracking through different processed frames.

Regarding tasks A and B, detection and tracking respectively, different implementation alternatives were tested.

5.2.1 Detection and tracking using image and video processing APIs from cloud computing providers

Initially, the use of image detection APIs from each cloud computing provider for task A and video processing APIs from each cloud computing provider for task B was considered.

To implement this approach, it was necessary to create image detection services in each of the providers (AWS, Azure, and GCP) and create video processing services for each of the providers (AWS, Azure, and GCP).

As presented in chapter 3, video processing services do not accept a video file as input, unlike image processing services, which directly accept the use of images as input (.jpg, .png files, for example, in bytes format).

To process a video, it is necessary to store it in the storage services of each provider, such as Azure Storage, Amazon S3, and Google Cloud Storage.

This adds a certain overhead to the processing flow, since it is first necessary to upload the video to these services to only then consume them and perform processing.

This approach also incurs additional costs, since there is a cost for storing videos in such storage services.

Tests revealed, however, a limitation of this approach, since the result of video processing APIs for task B do not return object tracking, but rather the list of objects detected through time. This problem does not allow determining unique object identifications and results in duplicate counting. For this reason this approach was abandoned.

5.2.2 Detection using image processing APIs and tracking processing in the cloud

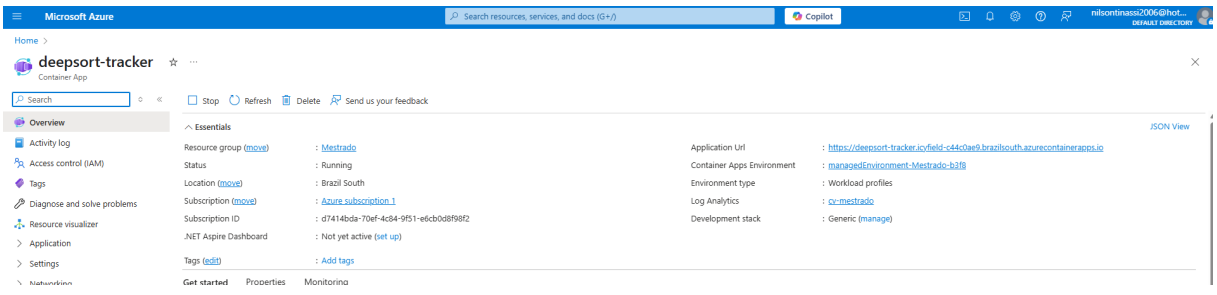
Alternatively, it was then decided to use exclusively the image processing API for both tasks (A - detection and B - tracking). However, due to the nature of such image detection services, they do not allow either video processing or object tracking. To circumvent these limitations, the second approach performs image detection for each video frame and then, in sequence, sends the detected frames together with the detection list for tracking processing using the DeepSort algorithm, as described in section 2.6 - DeepSORT (Deep Simple Online and Realtime Tracking).

This algorithm could not be implemented locally, since although functional, it would not allow evaluation of costs associated with task B - object tracking - for an application designed for cloud. The solution was to use cloud processing services made available by providers, such as Amazon App Run (subsection 3.5.1), Azure Container Apps (subsection 3.5.2), and Google Cloud Run (subsection 3.5.3).

These services, as presented in the indicated subsections, allow any device with internet access to send data to be processed in the cloud and billing is exclusively for time and resources used, unlike other solutions such as Amazon Fargate, Azure Container Instances, and Google Compute Engine whose pricing is per resource allocation[1] (regardless of the use of these resources).

In Figure 10 one can see the container app deepsort-tracker in Azure with some of its characteristics, such as the region to which it is configured (Brazil South), the current state of the service (running or stopped), and the access URL, that is, the endpoint that can be used to send data for processing.

Figure 10 – Azure Container App deepsort-tracker, capture from Azure Portal with main service information such as status and Application URL.



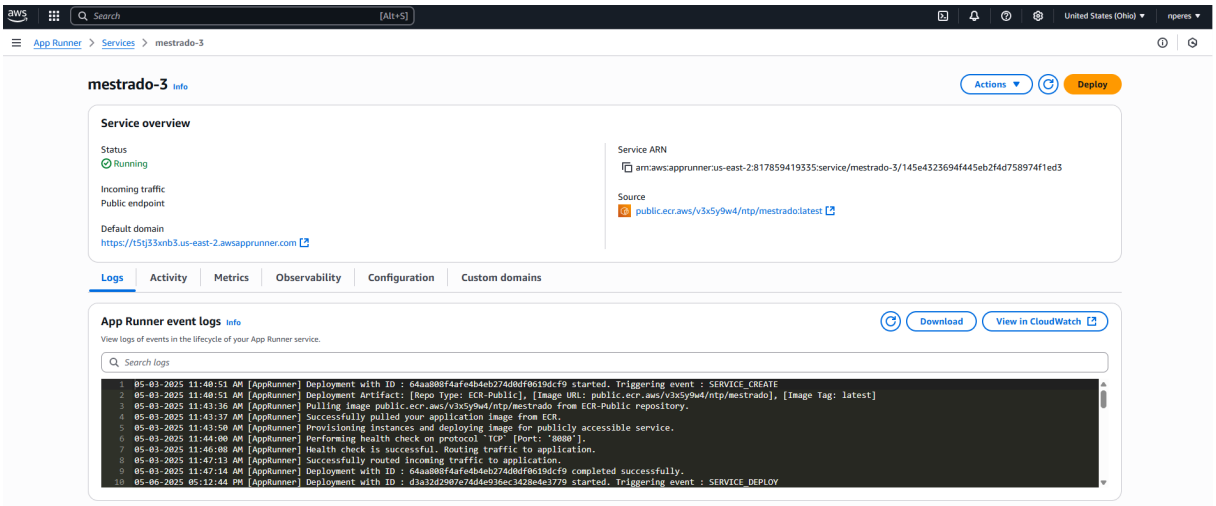
Source: Elaborated by the author.

In Figure 11 one observes the mestrado-3 service and also some relevant information

[1] It is very important to pay attention to the difference in architecture choice here, since choosing the wrong service for application execution can represent unwanted and unexpected costs.

about it, such as the service state (running or stopped), the application source (source code), and the URL or default domain for sending requests.

Figure 11 – Amazon App Run mestrado-3, capture from AWS resource management page with main service information such as status and Default domain.



Source: Elaborated by the author.

Finally, in Figure 12 is found the gcp-deepsort-tracker, the Google Cloud Run application, along with relevant information such as the URL for sending data for processing and the current service state (traffic and scale).

For all 3 services, the same source code was used for the DeepSort tracking algorithm and the same environment (same modules, libraries, and dependencies). This strategy was performed by creating an image using docker and then publishing this image on Dockerhub in public domain, as shown in Figure 13.

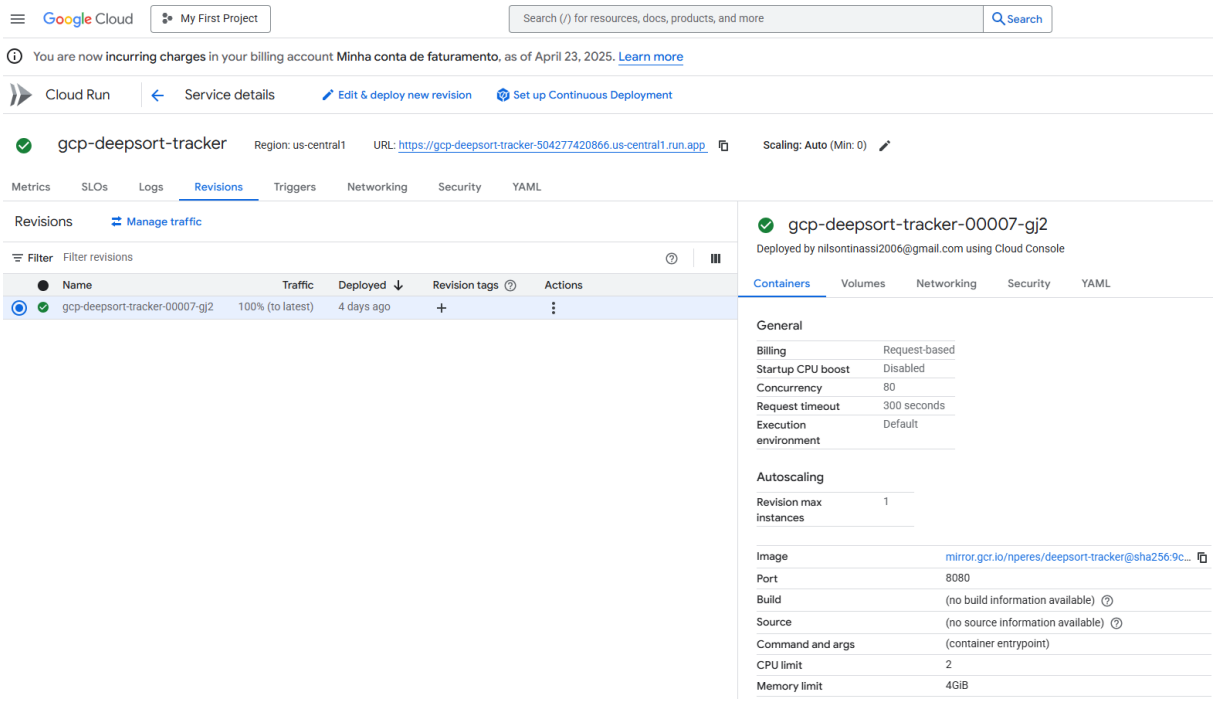
Both Azure Container Apps and Google Cloud Run have integration with Dockerhub and can read the image directly from this source, but Amazon App Run does not have this functionality, so the same image was then made available through another Amazon service, the Elastic Container Registry (ECR), as shown in Figure 14. It is important to note that such need to use ECR instead of Dockerhub results in additional costs associated with this service.

These applications developed for post-detection object tracking use the deep-sort-realtime library available for Python through the pip package manager, as shown in Figure 15.

The differential of the deep-sort-realtime library is the ease of coupling object tracking in applications that already perform object detection in images (allowing direct use of bounding boxes and confidences as input) and also support for real-time inference obtained through algorithm optimization compared to the original implementation, being suitable for applications with latency restrictions, such as embedded systems or edge computing scenarios.

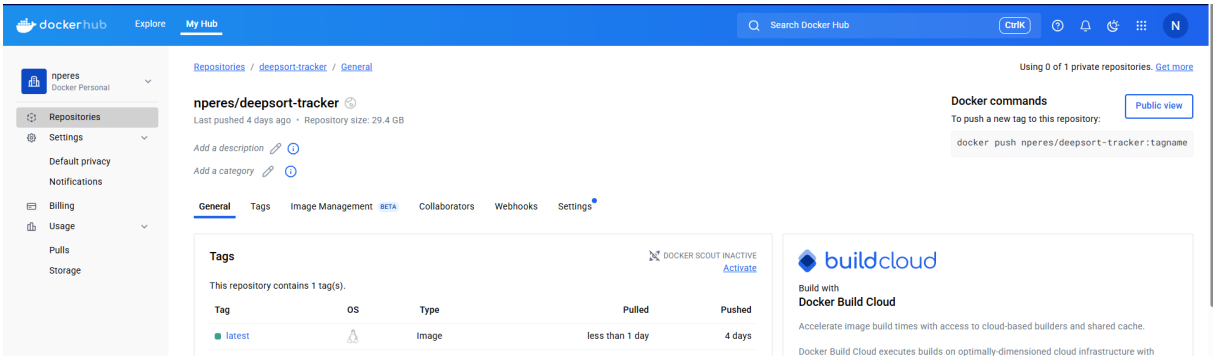
The cloud processing services, Azure Container App, Amazon App Run, and Google

Figure 12 – Google Cloud Run gcp-deepsort-tracker, capture from GCP resource management page with main service information such as status and URL.



Source: Elaborated by the author.

Figure 13 – Screenshot of Dockerhub repository with the image used for the DeepSort algorithm.



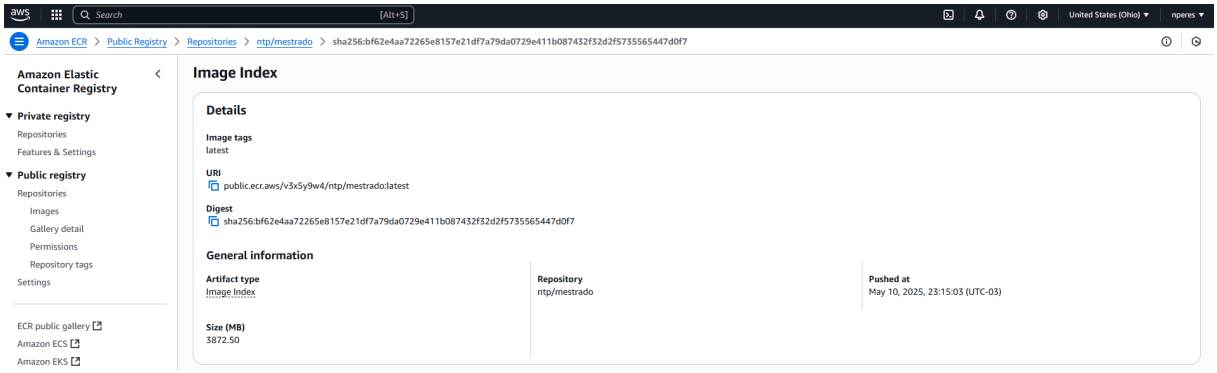
Source: Elaborated by the author.

Cloud Run require as part of the scope of their respective configurations that the resource values that will be available to the application be defined.

For the 3 applications, the same resources were defined to allow more adequate comparison in terms of performance. The balancing factor to determine the amount of RAM memory and CPU resources to be made available to the application was the available edge processing device (Raspberry Pi model 4), which has 4 GB of RAM memory and 2 available CPUs (Table 17).

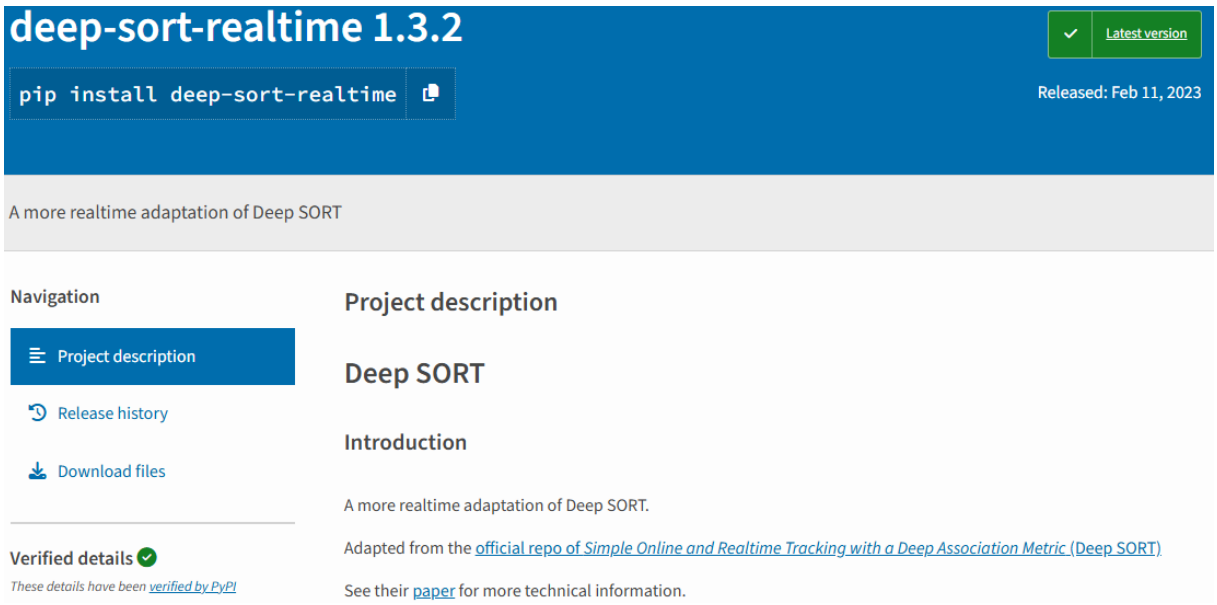
Thus, 4 GB of memory and 2 CPUs were also selected for each of the applications,

Figure 14 – Screenshot of Amazon's proprietary container repository (ECR).



Source: Elaborated by the author.

Figure 15 – Screenshot of pip package manager documentation for Python indicating the deep-sort-realtime library used in the solution.



Source: Elaborated by the author.

whether in Azure, AWS, or Google[2].

Given that one of the central objectives of the project is cost analysis and comparison, it was necessary to monitor the execution cost of each object tracking application for each of the videos. For this monitoring, it was necessary to activate specific monitoring resources for these applications such as Azure Cloudwatch, Amazon Logs, and Google Cloud Logs.

This was the implementation used for validation and experimentation for object detection

[2] NOTE: Processing prices are proportional to the amount of resources made available and the use of these resources, meaning that making more resources available (memory and CPUs) can represent both a performance gain and an increase in costs.

and tracking using solutions made available by cloud computing providers.

5.2.3 Detection and tracking on the edge processing device

For edge video processing, the most recent version of the Yolo model (v11) nano was used with the objective of better understanding processing limitations with limited resources. The device chosen for this was the Raspberry Pi model 4 with 4 GB of RAM memory, as shown in Table 17.

Since the computational power of the edge device is already limited by itself, the access interface for sending videos to be processed was hosted on a local computer with effects of simulating server behavior. Communication between the Raspberry and the local computer (server) occurs through the wi-fi network, provided that both are connected to the same network.

Table 17 – Main specifications of Raspberry Pi 4 Model B 4GB.

Specification	**Raspberry Pi 4 Model B 4GB**
Processor	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz
RAM Memory	4GB LPDDR4-3200 SDRAM
Storage	microSD 64GB
Connectivity	Gigabit Ethernet, Wi-Fi 802.11 b/g/n/ac, Bluetooth 5.0
USB Ports	2x USB 3.0, 2x USB 2.0
Video Outputs	2x micro-HDMI (up to 4Kp60)
Power	USB-C 5V/3A
Operating System	Raspberry Pi OS (Linux)
Dimensions	85.6 mm x 56.5 mm

Source: Foundation (2024).

5.3 Metrics for result evaluation

The calculation of metrics occurs automatically as a consequence of measurements taken during application runtime.

Latency (L) is measured as the time between sending the processing request and receiving the response from each segment, being expressed in milliseconds (ms). In the context of task A, where individual images or video frames are processed separately, latency is calculated for each of the extracted frames. The final result of this metric is obtained through the arithmetic mean of latencies measured for each frame, according to Equation 5.1.

$$L = \frac{1}{N}\sum_{i=1}^{N}(t_{response,i} - t_{send,i}) \tag{5.1}$$

Where:

- N = total number of frames analyzed
- $t_{send,i}$ = sending instant of frame i
- $t_{response,i}$ = response receiving instant of frame i

Complete processing time (T_p) is measured as the total time required for processing the entire video in task B, considering from the beginning of sending until receiving the final result, according to Equation 5.2.

$$T_p = t_{end} - t_{start} \tag{5.2}$$

Where:

- t_{start} = initial instant of video sending
- t_{end} = instant of final result reception

The number of frames processed per second or throughput (FPS) is calculated from the total number of video frames (N_f) divided by the total processing time, according to Equation 5.3.

$$FPS = \frac{N_f}{T_p} \tag{5.3}$$

In many machine learning applications, recall and precision metrics are used, whose calculation involves knowing exactly the amount of true positives, true negatives, false positives, and false negatives.

However, in the proposed application, which involves object tracking in video, the use of these metrics becomes problematic. This is because it is difficult to precisely determine which objects should be considered relevant (that is, what is the "ground truth" of unique instances in movement), especially in scenarios with occlusions, partial entries and exits from the field of view, or objects that appear only briefly.

Furthermore, the definition of what constitutes a false positive or a false negative can be ambiguous: for example, the same object can be detected and tracked with multiple different identifiers over time (fragmentation), or different objects can be erroneously tracked as if they were one (identity swap).

These limitations make exact counting of TP, FP, and FN difficult, making the application of precision and recall impractical in practice.

As an alternative, the use of absolute percentage error between the expected count of unique objects and the count effectively obtained was chosen. For example, if 3 distinct cars were expected in a video and only 2 were correctly tracked, the error is 33.3

In addition to performance metrics, a cost metric associated with analysis execution in each architecture was also considered. For the cloud environment, cost was calculated based on total processing time, the value charged per minute of execution, and cost per processed image, according to the pricing models of the service used.

For the edge environment, cost was estimated based on execution time and energy consumption of the device, considering the nominal power of the hardware.

Calculations were performed independently for each architecture, without any average or combination between results.

In this way, costs for either cloud or edge environment were estimated through cost parameters provided by each provider, according to Table 18. This table presents the unit values used for cost calculations per video in each environment, considering computational resources (CPU and memory), number of requests, and, when applicable, specific rates per image detection. In the case of the edge solution, cost was estimated based on energy consumption, hardware depreciation, and additional monthly costs.

Table 18 – Cost parameters by cloud service provider (Azure, AWS, GCP) and for edge processing for detection and tracking execution.

Parameter	**Azure**	**AWS**	**GCP**	**Edge**
vCPU (per second)	$0.000024	NA	$0.000024	$0.00000105
Memory (per GB/s)	$0.000003	NA	$0.0000025	NA
Cost per 1M requests	$0.40	NA	$0.40	NA
Idle vCPU (per second)	$0.0000036	NA	NA	NA
Free tier (vCPU s / Mem GB.s / Reqs / Egress GB)	0 / 0 / 2M / NA	NA	0 / 0 / 2M / 1	NA
Detection per image	$0.001	$0.001	$0.0015	NA
Configuration (vCPU / GB)	NA	2 / 4	NA	NA
Active vCPU (per hour)	NA	$0.064	NA	NA
Memory (per GB/hour)	NA	$0.007	NA	NA
Minimum active billing	NA	1 minute	NA	NA
Monthly energy	NA	NA	NA	$0.54
Monthly hardware	NA	NA	NA	$1.67
Monthly overhead	NA	NA	NA	$0.50
Total monthly cost	NA	NA	NA	$2.71

Source: Elaborated by the author.

Note: Fields indicated by NA (not applicable) indicate that this parameter is not part of the corresponding provider's pricing structure.

5.4 Video samples for validation and experimentation

Among the experimentation objectives, evaluation of solution robustness under varied conditions stands out. Therefore, the videos used for testing were recorded according to the

conditions presented in Table 19.

Table 19 – Duration, weather, time, and location characteristics by video used in experimentation.

ID	Duration	Weather	Time	Location
1	1:00	Day / rain	08:19	R. Ana Luiza, 260
2	2:00	Day / sun	12:46	Av. 9 de Julho, 217
3	3:00	Day / clouds	17:57	Av. Tancredo Neves, 505
4	3:00	Night / rain	18:35	R. Matheus Marinelli, 18
5	1:00	Day / sun	17:52	R. Santos di Lello, 217
6	2:00	Night / rain	18:43	Av. João Luís de Oliveira, 29
7	3:00	Night / rain	18:30	Av. Ayrton Senna, 800

Source: Elaborated by the author.

Note[1]: All locations refer to the city of Batatais, São Paulo, Brazil.

Note[2]: All videos were captured with 720p resolution and 30 FPS.

All experiments used the same input videos, captured in a controlled environment, ensuring parity between evaluated scenarios.

5.5 Application flow

The user accesses the main application endpoint; in the case of the experiment, the application was configured for local execution, as shown in Figure 16.

On this page, the user must click on "Select Video", then a pop-up window will open and allow the user to select the video for processing, as in Figure 17.

Next, the user must fill in the text boxes "Expected Vehicles" and "Expected People", as in Figure 18. This information is used by the application to process absolute percentage error results.

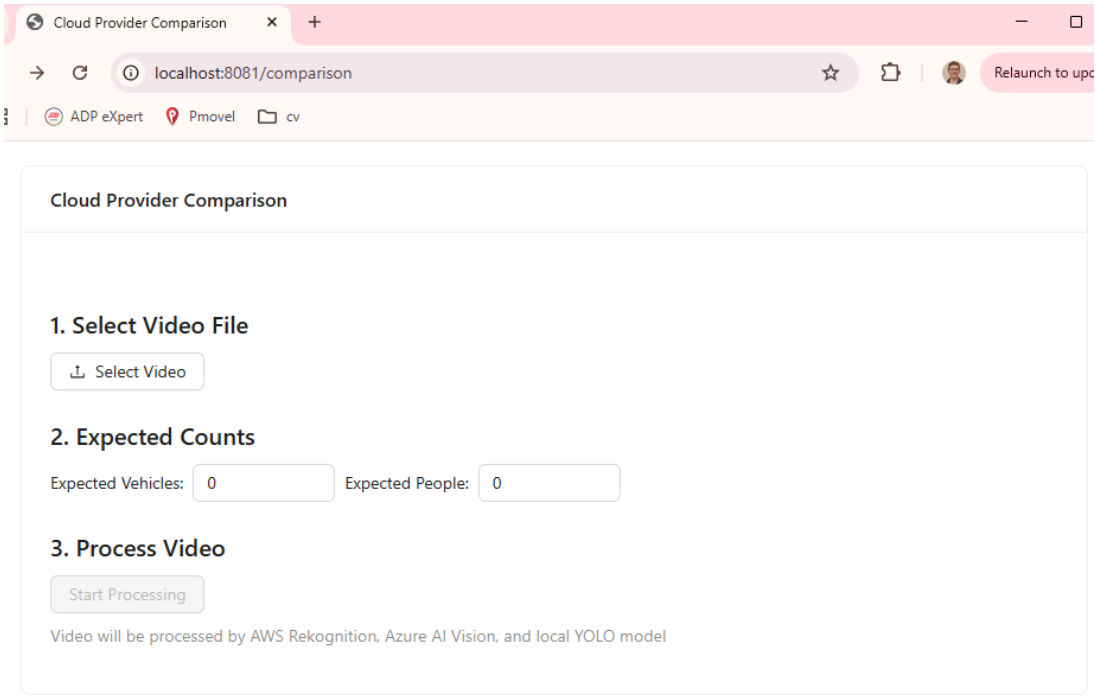
Finally, the user must select the "Start Processing" option.

When doing this, the backend will send a request to the application responsible for processing called "ml-service". But before sending the request, it is necessary to check if the ml-service application is active through a request to the /health endpoint. If affirmative, the video upload that will be processed is performed. Only after upload completion is a request sent to start processing. These details can be observed in the backend container logs in Figure 19.

The ml-service then receives the video upload and immediately afterward also receives the request to start processing (Figure 20).

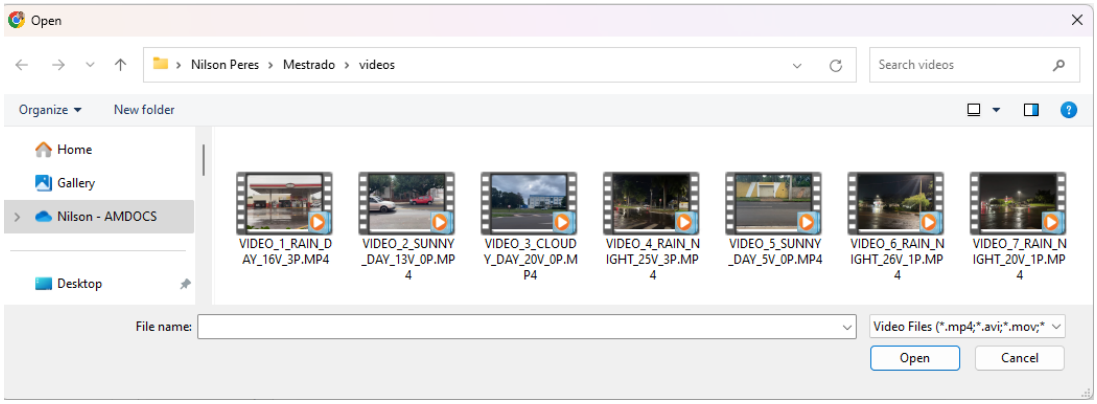
The first step of processing is to divide the video into frames and save these in a repository; this strategy facilitates result validation.

Figure 16 – Screenshot of the front-end (page that allows user interaction with the service) displaying the application's initial page indicating the inputs that must be provided to start processing.



Source: Elaborated by the author.

Figure 17 – Screenshot of the front-end (page that allows user interaction with the service) displaying video selection for processing.



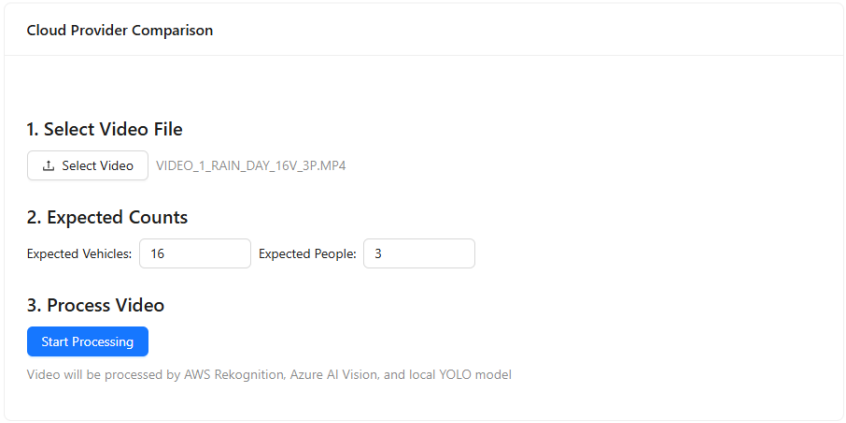
Source: Elaborated by the author.

Then for each of the providers, processing begins which is done in 2 loops, by provider and by frame. This allows accounting for processing time for each provider independently.

The logs (Figure 21) show some of the captured information and also indicate the processing flow.

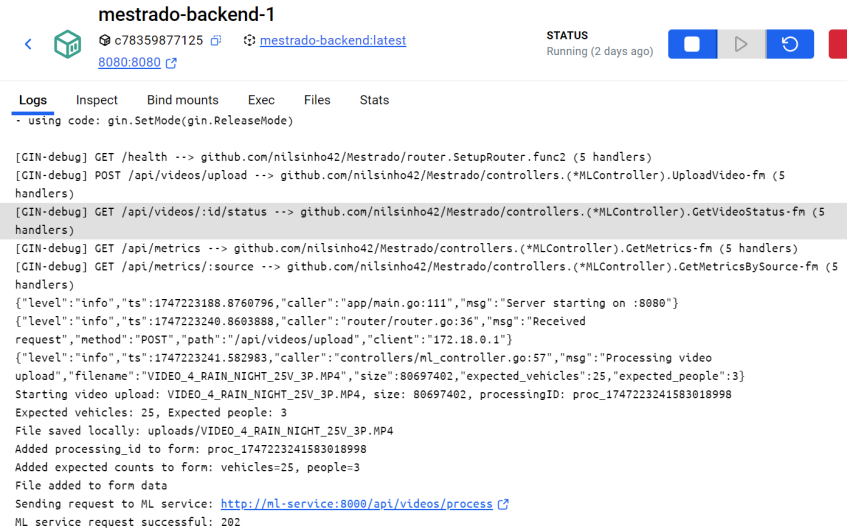
For example, the indicator [aws] determines for which provider processing is being performed, then the next indicator [1813] determines which frame is being processed. The "Latency" information determines the latency time for image processing in the referred provider's

Figure 18 – Screenshot of the front-end (page that allows user interaction with the service) displaying the expected quantities of people and vehicles fields filled in.



Source: Elaborated by the author.

Figure 19 – Backend container - sending requests to ml-service



Source: Elaborated by the author.

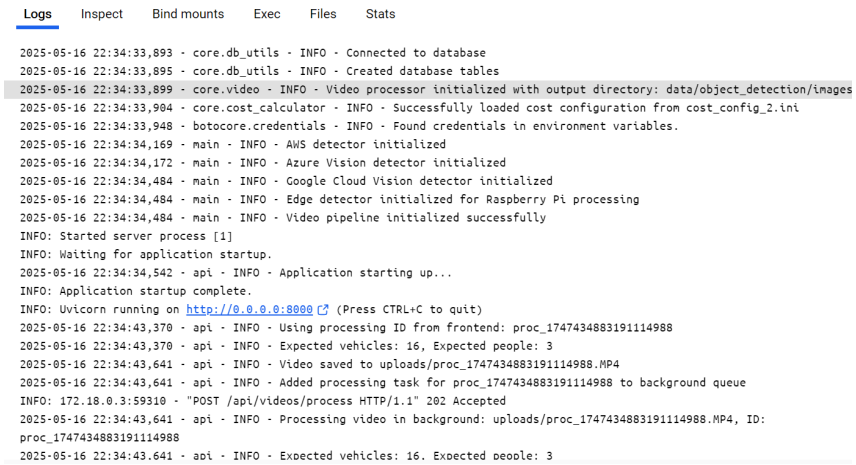
detection API.

After receiving detections from the provider, preprocessing is performed which has 2 objectives: 1) filter detections that do not belong to target classes (people/vehicles) and 2) correct rectangle/box coordinates to a common standard for all providers.

Finally, the tracking and unique object identification stage is performed using the Deep-SORT algorithm. This stage is performed according to the environment; in the case of each cloud computing provider, the associated solution is used (AWS App Run, Azure Container App, or Google Cloud Run) and for edge computing, the service available on the Raspberry Pi itself is used.

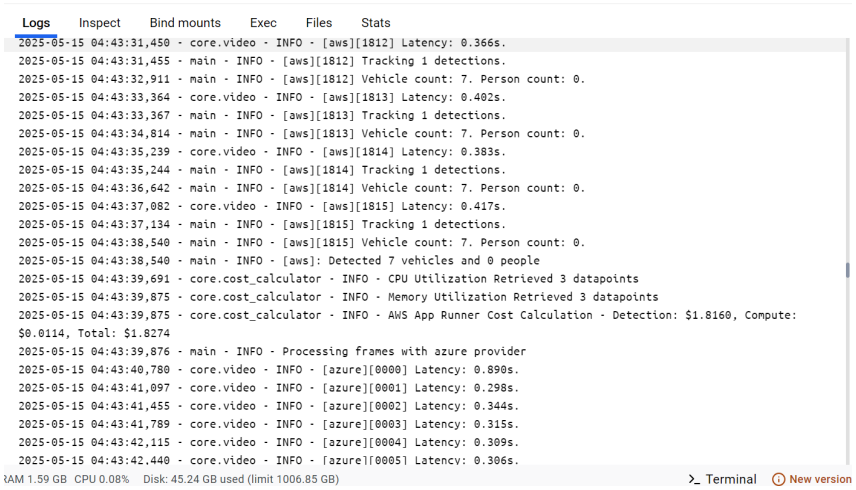
The complete code of the developed applications is available for public consultation

Figure 20 – ml-service - receiving processing requests



Source: Elaborated by the author.

Figure 21 – Processing logs of ml-service indicating vehicle and people counting during processing time, capture taken from docker container terminal.



Source: Elaborated by the author.

through the online repository: <https://github.com/nilsinho42/Mestrado>.

CHAPTER

6

RESULTADOS

The current section aims to present and discuss the results obtained from the execution of the object detection and tracking application, using both cloud solutions and edge processing. The analysis is structured based on multiple previously defined evaluation criteria, allowing comparison of different environments under various perspectives relevant for technical and economic decision-making. Among the evaluated components, processing cost stands out as a critical factor, especially for large-scale applications or continuous operation. The results are organized by evaluation category — cost, latency, detection and tracking quality, among others — allowing for a detailed comparative analysis between the studied approaches.

6.1 Processing costs

The results in dollars (standard billing currency of cloud service providers) for the analysis cost of each of the 7 tested videos are presented in Table 20. The parameters for cost

Table 20 – Estimated costs in USD per video for object detection and tracking for each cloud service provider and for edge processing.

ID	Weather	Duration	Location	Azure Cost	AWS Cost	GCP Cost	Edge Cost
1	Day / rain	1:00	R. Ana Luiza, 260	$ 1.90	$ 1.86	$ 2.79	$ 0.0034
2	Day / sun	2:00	Av. 9 de Julho, 217	$ 3.65	$ 3.64	$ 5.46	$ 0.0058
3	Day / clouds	3:00	Av. Tancredo Neves, 505	$ 5.43	$ 5.43	$ 8.14	$ 0.0081
4	Night / rain	3:00	R. Matheus Marinelli, 18	$ 5.43	$ 5.43	$ 8.14	$ 0.0081
5	Day / sun	1:00	R. Santos di Lello, 217	$ 1.84	$ 1.84	$ 2.75	$ 0.0028
6	Night / rain	2:00	Av. João L. de Oliveira, 29	$ 3.69	$ 3.67	$ 5.50	$ 0.0058
7	Night / rain	3:00	Av. Ayrton Senna, 800	$ 1.84	$ 1.83	$ 2.73	$ 0.0027

Source: Elaborated by the author.

calculation were introduced in section 5.3. The costs presented in Table 20 are estimated as

follows:

$$T_{\text{total}} = C_{\text{det}} \cdot n + T_{\text{active}} \cdot (q_{\text{CPU}} \cdot C_{\text{CPU}} + q_{\text{mem}} \cdot C_{\text{mem}}) + T_{\text{idle}} \cdot (q_{\text{mem}} \cdot C_{\text{mem}}) \quad (6.1)$$

Where:

$C_{\textbf{det}}$: Object detection API request cost per image;

$T_{\textbf{active}}$: Application usage time for object tracking per image;

$T_{\textbf{idle}}$: Time containers remain in idle state;

$C_{\textbf{mem}}$: Cost per second per GB of RAM memory;

$C_{\textbf{CPU}}$: Cost per second per CPU;

$q_{\textbf{mem}}$: Amount of memory allocated in app/container (in GB);

$q_{\textbf{CPU}}$: Amount of CPU allocated in app/container;

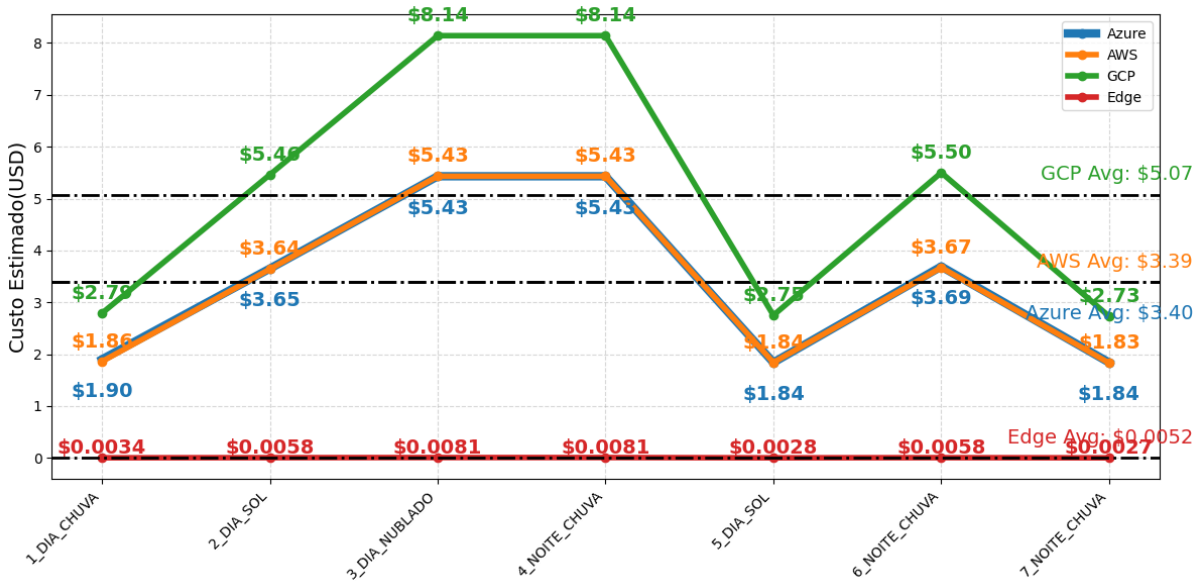
$p_{\textbf{mem}}$: Average percentage of memory usage during processing;

$p_{\textbf{CPU}}$: Average percentage of CPU usage during processing;

n: Number of frames in the video.

That is, the estimated detection cost is determined by the number of frames to be processed by the detection API (n) multiplied by the detection cost of each provider (C_{det}). This result must then be summed with the tracking cost, which is determined by the processing time for the complete video (T_{active}) multiplied by the total cost of allocated CPU ($q_{CPU} \cdot C_{CPU}$) and by the cost of total allocated memory ($q_{mem} \cdot C_{mem}$). Since the applications were configured to always scale to zero during idle periods, the cost associated with these periods determined by $T_{idle} \cdot (q_{mem} \cdot C_{mem})$ is null. The data from Table 20 are also presented in graph format in Figure 22. From this graph, it is observed that the behavior of the cost curve for different cloud service providers was the same, so that the curves follow the same characteristics, growing for longer duration videos and decreasing for shorter duration videos. This result is expected since the cost of cloud processing is proportional to video duration (the longer the video, the more frames to be processed and more processing time required). Another reason why this behavior is expected is that, despite being competitors and having different nuances, the main pricing structures of cloud service providers are quite similar to each other. It is also observed from this graph that the Azure and AWS curves are practically indistinguishable from each other, meaning that the cost of one follows the cost of the other, even for different videos. Another factor is the displacement of the curve representing GCP cost relative to the others. Despite following the same behavior of rises and falls, there is a displacement that results from the higher cost of the image detection API. While Azure and AWS charge $1.00 for every 1000 processed images, GCP charges 1.50$. This

Figure 22 – Costs in vehicle and people detection and tracking for each computing provider and for edge processing per analyzed video.



Source: Elaborated by the author.

significant 50It should be noted that the conditions under which the videos were captured - day or night, sun or rain - have no influence factor on the processing cost parameter. Finally, it is noted that the costs associated with edge processing are minimal compared to cloud processing costs. This difference is due to the fact that in edge processing there is no charge for computation time or use of third-party APIs.

6.2 Quality of detection, tracking and counting

In addition to processing cost, there are other very relevant factors for making a decision regarding which approach is most appropriate. One of these factors is the quality of the solution, i.e., whether the results obtained were close to expected or not. Considering the application's objective - counting vehicles and people using object detection and tracking solutions - the defined target result was to obtain correct counts of vehicles and people for each analyzed video. These results are presented in Table 21 and Table 22. The expected count of people and vehicles was determined from visual analysis of the video. A preliminary analysis of Table 21 reveals that the videos used for experimentation and validation presented few people - 3 videos have 0 people as expected count and another 2 videos have only 1 expected person. This small number of people is a limitation of the analysis. On the other hand, Table 22 presents a considerably higher expected count for vehicles, allowing for a more in-depth analysis of the results. The visualization of these table results is facilitated through the graphs in Figure 23 and Figure 28. Figure 23 displays the bar graphs of count per provider versus expected counts per experiment video.

Table 21 – People counting results in object detection and tracking for each cloud service provider and for edge processing per video.

				Count				
ID	Weather	Duration	Location	Azure	AWS	GCP	Edge	Expected
1	Day / rain	1:00	R. Ana Luiza, 260	3	3	2	2	3
2	Day / sun	2:00	Av. 9 de Julho, 217	2	1	4	2	0
3	Day / clouds	3:00	Av. Tancredo Neves, 505	0	0	0	0	0
4	Night / rain	3:00	R. Matheus Marinelli, 18	1	1	1	1	3
5	Day / sun	1:00	R. Santos di Lello, 217	0	0	0	0	0
6	Night / rain	2:00	Av. João L. de Oliveira, 29	1	2	1	1	1
7	Night / rain	3:00	Av. Ayrton Senna, 800	0	0	0	0	1

Source: Elaborated by the author.

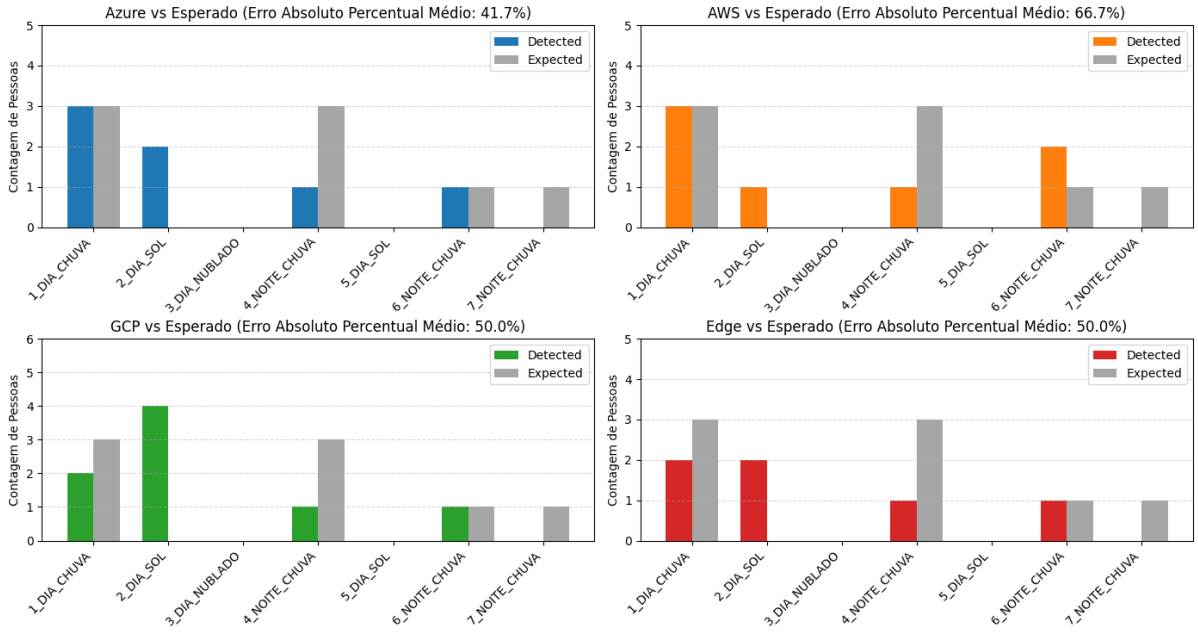
Table 22 – Vehicle counting results in object detection and tracking for each cloud service provider and for edge processing per video.

				Count				
ID	Weather	Duration	Location	Azure	AWS	GCP	Edge	Expected
1	Day / rain	1:00	R. Ana Luiza, 260	13	15	22	18	16
2	Day / sun	2:00	Av. 9 de Julho, 217	22	24	13	8	13
3	Day / clouds	3:00	Av. Tancredo Neves, 505	3	3	3	1	20
4	Night / rain	3:00	R. Matheus Marinelli, 18	4	7	5	6	25
5	Day / sun	1:00	R. Santos di Lello, 217	6	5	6	6	5
6	Night / rain	2:00	Av. João L. de Oliveira, 29	17	19	17	13	26
7	Night / rain	3:00	Av. Ayrton Senna, 800	3	7	5	3	20

Source: Elaborated by the author.

Considering first the results obtained for Azure (upper left quadrant of Figure 23), it is observed that in videos 1_DIA_CHUVA and 6_NOITE_CHUVA it was possible to determine exactly the expected number of people (3 and 1, respectively) and that in videos 3_DIA_NUBLADO and 5_DIA_SOL the obtained count of 0 people was as expected. Analyzing the results of video 1_DIA_CHUVA in more detail, it is perceived that of the 3 people identified, 2 were true positives (correctly identified), 1 was false positive (duplicate identification) and 1 false negative (not identified) - see Figure 24.

Figure 23 – Comparison of people detection and tracking results for each cloud service provider and for edge processing with the expected people count value per video.



Source: Elaborated by the author.

Figure 24 – Detailed investigation of people counting for video 1_DIA_CHUVA for Azure cloud service provider indicating 2 examples of true positives, 1 example of false positive and 1 example of false negative.

(a) True positive 1 - In an image containing one person, the analysis found one person and identified it uniquely, indicating correct processing.

(b) True positive 2 - In an image containing two people, the analysis found one person and identified it uniquely. The other person had already been identified in previous video frames.

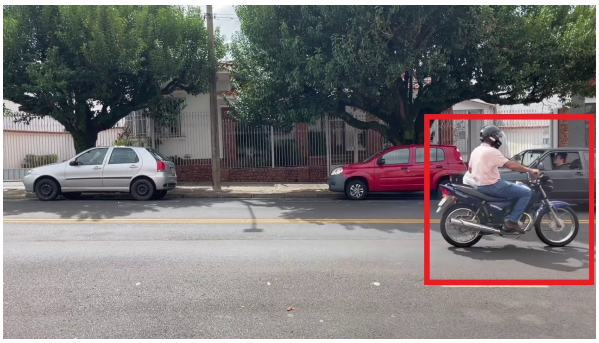
(c) False positive 1 - In an image containing two people, the analysis found a person that had already been identified, resulting in duplication.

(d) False negative 1 - In an image containing one person, the analysis found no person, resulting in processing error.

Source: Elaborated by the author.

For video 2_DIA_SOL, no person detection was expected, yet 2 were detected (false positives). As Figure 25 reveals, Azure's image detection API in some cases detects motorcycles as people, leading to classification confusion that affects both people counting (false positives) and vehicle counting (false negatives).

Figure 25 – Detailed investigation of people counting for video 2_DIA_SOL for Azure cloud service provider indicating 2 examples of false positives related to identification of drivers (people) in vehicles.



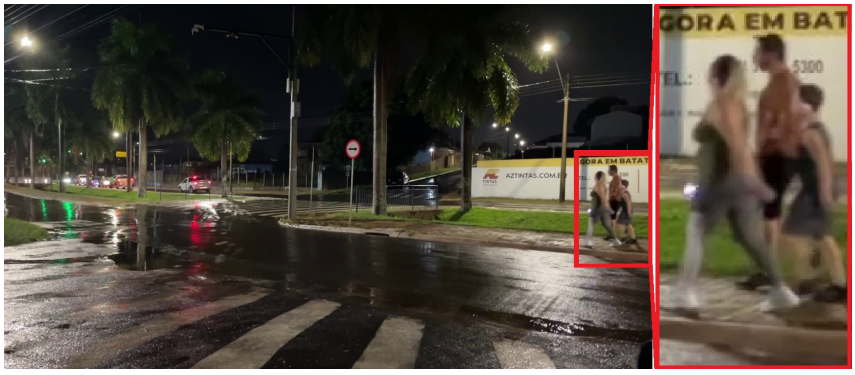
(a) False positive 1 - In an image with multiple vehicles but no people, the driver of a vehicle is identified as a person.

(b) False positive 2 - In an image with multiple vehicles but no people, the driver of a vehicle is identified as a person.

Source: Elaborated by the author.

For video 4_NOITE_CHUVA, it was observed that 3 people should be counted in the video, but the analysis result was only 1 person. A more in-depth analysis as in Figure 26 shows that the 3 people were very close to each other, so that during processing this group was considered as a single person. A similar behavior was observed in processing using Amazon

Figure 26 – Detailed investigation of people counting for video 4_NOITE_CHUVA for Azure cloud service provider indicating 1 example of false positive related to identification of 1 person in an image where 3 people should be identified.

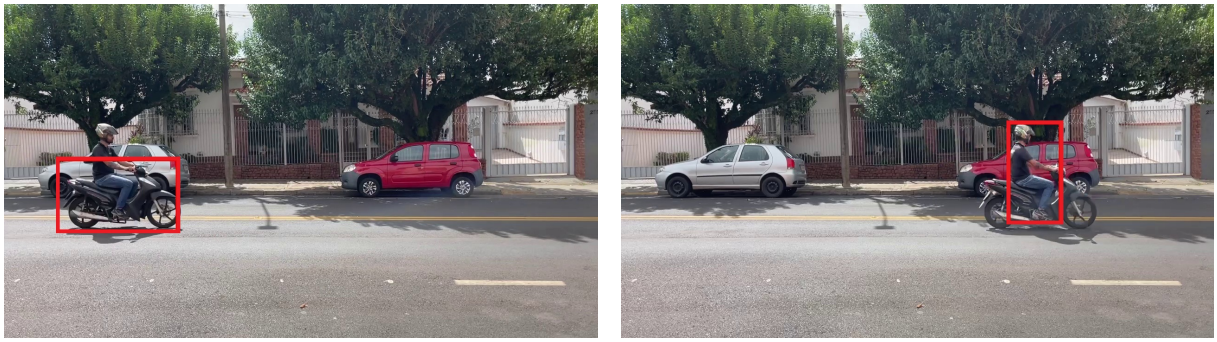


Source: Elaborated by the author.

AWS solutions (upper right quadrant of Figure 23). For example, videos 3_DIA_NUBLADO and 5_DIA_SOL correctly obtained the count of 0 people. Video 1_DIA_CHUVA also obtained

the correct count, but like Azure processing, counted the same person twice and left another out (1 false positive and 1 false negative). For video 4_NOITE_CHUVA the same behavior as Azure was also observed, where the group of 3 people was identified as only 1. In video 2_DIA_SOL a curious fact happened, where one of the motorcycles was correctly classified as a vehicle, but for the other motorcycle there were 2 classifications, as shown in Figure 27. The same also happened in processing video 6_NOITE_CHUVA.

Figure 27 – Detailed investigation of people counting for video 2_DIA_SOL for AWS cloud service provider indicating 1 example of true negative and 1 example of false positive related to identification of drivers (people) in vehicles.



(a) True negative - In an image with a driver and a vehicle, but no isolated person, only the vehicle is identified.

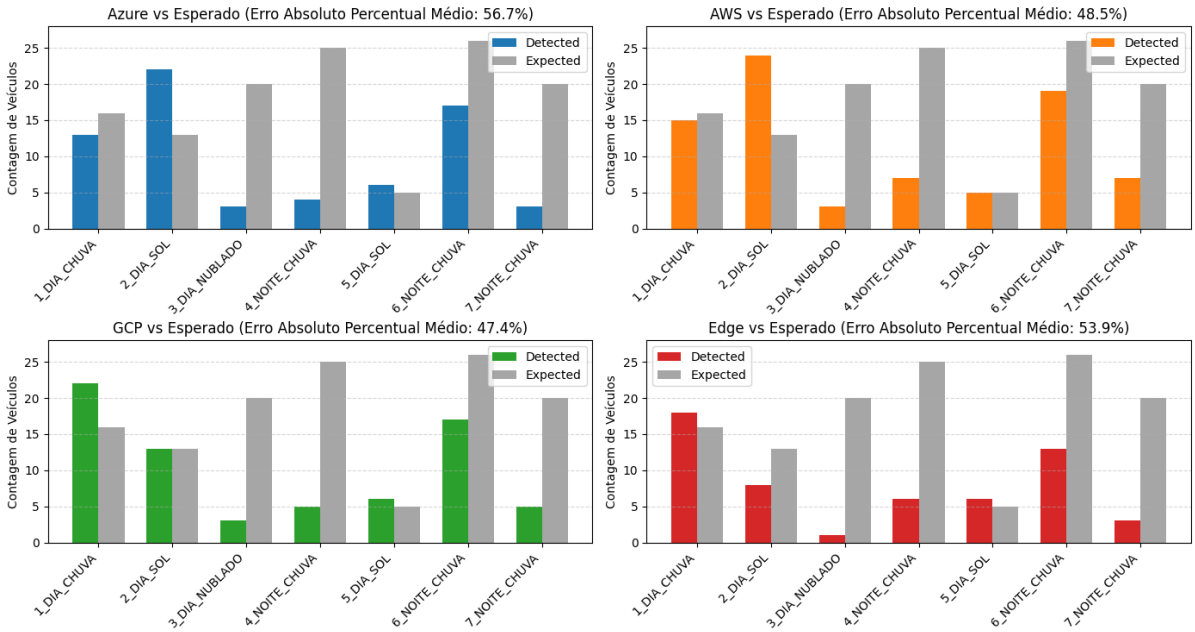
(b) False positive - In an image with a driver and a vehicle, but no isolated person, the driver is identified as a person.

Source: Elaborated by the author.

For processing results using GCP (lower left quadrant of Figure 23) the variations were somewhat larger. The similarity lies in videos 3_DIA_NUBLADO and 5_DIA_SOL which correctly identified 0 people as expected, in video 4_NOITE_CHUVA which also identified the group of 3 people as being only 1, and in video 6_NOITE_CHUVA which also identified the motorcycle driver as 1 person. However, for video 1_DIA_CHUVA, Google processing counted only 2 people instead of the expected 3. To make matters worse, the count of 2 was incorrect, since it counted the same person twice (1 false positive) and left 2 people out (2 false negatives). And for video 2_DIA_SOL, GCP processing was by far the one with the greatest error, counting 4 people when none were expected - where of the 4 people counted, 2 were the drivers of the 2 motorcycles that appear in the video. This behavior also reveals a difficulty in motorcycle classification by Google's detection API. Finally, edge processing analysis (lower right quadrant of Figure 23) reveals that for video 1_DIA_CHUVA, edge processing failed to count 1 of the people (count of 2 versus 3 expected). However, the 2 people identified are correct (no false positives). The false negative (third individual who was not included in the count) can be explained by the small size of the individual in the image (edge processing detection uses the YOLO model which has analysis limitations for small objects). Analysis of videos 3_DIA_NUBLADO and 5_DIA_SOL correctly resulted in 0 people. And the count of videos 2_-

DIA_SOL and 4_NOITE_CHUVA erroneously classified motorcycles as people (false positives). A similar analysis can be conducted on vehicle counting results (Figure 28). However, since expected vehicle counts are higher, deeper analysis in terms of vehicle-by-vehicle evaluation is compromised. For all providers, including edge processing, a very high error is observed during

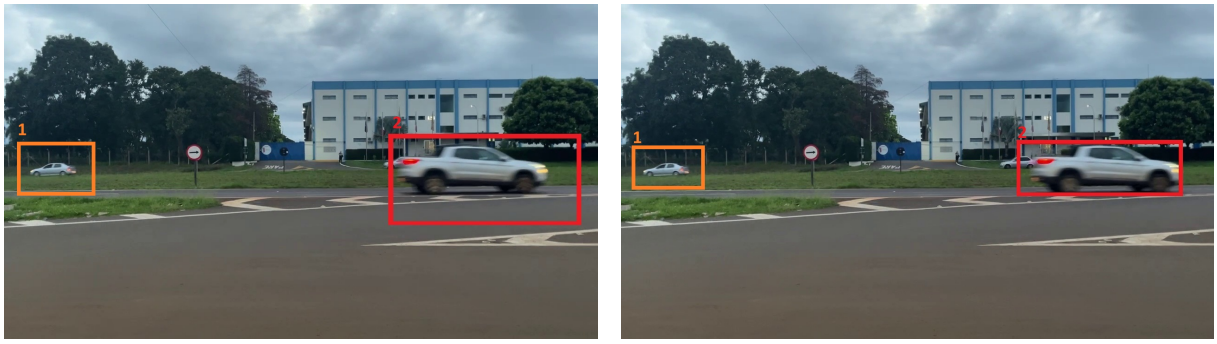
Figure 28 – Comparison of vehicle detection and tracking results for each cloud service provider and for edge processing with the expected vehicle count value per video.



Source: Elaborated by the author.

processing of videos 3_DIA_NUBLADO, 4_NOITE_CHUVA and 7_NOITE_CHUVA. A more in-depth analysis reveals that video 3_DIA_NUBLADO involves filming vehicles traveling on a highway with higher traffic and where vehicles transit at higher average speed. Additionally, the capture distance from vehicles traveling in the farthest lane makes these vehicles (objects) appear smaller in the video. Both these characteristics can be observed in Figure 29.

Figure 29 – Example of capture and tracking - 3_DIA_NUBLADO - background elements represented in smaller size due to capture distance, elements are distorted due to traffic speed. Both images show a true positive (box 2, in red) and a false negative (box 1, in orange). The vehicle identified by box 2 (red) was identified by processing, but the vehicle identified by box 1 (orange) was not identified.



(a) Processing performed on Azure.

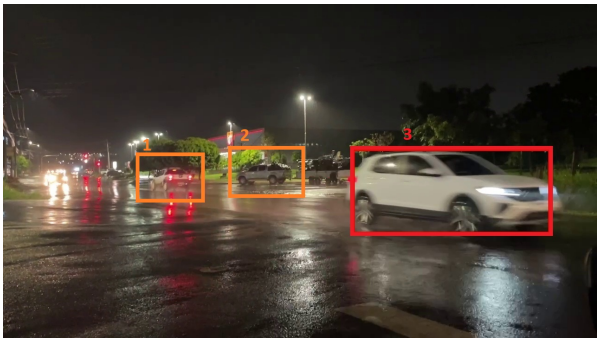
(b) Processing performed using GCP.

Source: Elaborated by the author.

For video 4_NOITE_CHUVA, the high error also results from the distance between objects/vehicles to the capture point. The capture of this video was performed at a diagonal angle and comprises both directions of an arterial road (speed up to 60 km/h), where the opposite direction of the road was at a considerably more distant point from the capture point, since there is separation by a river that divides the directions of this road, as presented in Figure 30a. In this video, it was observed that if the expected count only considered one direction of the road, the expected number of vehicles would be only 10 and not 25. Considering this new value as target, the result would already be substantially closer than what was originally obtained. Still, the error remains significant — Azure: 4, AWS: 7, GCP: 5, compared to the new expected value of 10. A similar error behavior is observed in video 7_NOITE_CHUVA, which is also presented in Figure 30b. In this case, it was also not possible to identify any of the vehicles that transited in the opposite direction of the road. If the expected count considered this factor, the updated expected number would be 7 vehicles instead of 20, a result that would be closer to what was obtained (Azure 3, AWS 7, GCP 5 and Edge 3) than the original.

Figure 30 – Example of capture and tracking for 4_NOITE_CHUVA and 7_NOITE_CHUVA.

(a) GCP - 4_NOITE_CHUVA - In an image with 3 vehicles, the analysis was able to correctly identify 2 vehicles (boxes 2 and 3 in red), but failed to identify 1 vehicle (box 1 in orange), resulting in 2 true positives and 1 false negative.



(b) Edge - 7_NOITE_CHUVA - In an image with 3 vehicles, the analysis was able to correctly identify 1 vehicle (box 3, in red), but failed to identify 2 vehicles (boxes 1 and 2, in orange), resulting in 1 true positive and 2 false negatives.

Source: Elaborated by the author.

Overall, the behavior of results across different cloud service providers is similar, which is due to the fact that all use the same *deepsort* algorithm. Since each provider uses distinct computer vision APIs, with different models, *thresholds* and inference strategies, it is expected that the initial number of detections varies, directly impacting the total count of tracked objects. Regarding *deepsort* parameterization, the Azure model uses a higher *nn_budget* (300), which tries to compensate for the lower number of detections, but operates with more restrictive similarity (*max_cosine_distance* = 0.4) and overlap (*max_iou_distance* = 0.7) *thresholds*, which may limit the ability to form and maintain tracks for objects detected under unfavorable visual conditions. GCP adopts more permissive parameters (*max_cosine_distance* = 0.85 and *max_iou_distance* = 0.75), which facilitates track creation, even if some of them result from detections with lower confidence — thus favoring a higher number of detections and, consequently, a higher risk of false positives. This operational difference between configurations reflects in the variations observed in object counting between providers. Regarding variable video capture conditions (lighting, angle, distance, speed of movement of the object to be tracked, among others), adjusted results are defined in relation to expected quantities (target values) of vehicle counting. These adjusted results allow for better comparison between different capture conditions to determine how these conditions influence the final processing result. These adjusted results together with absolute and percentage error calculations are demonstrated in Table 23, Table 24, Table 25 and Table 26. The results of Table 23 show that all cloud providers presented performance relatively close to expected, with AWS standing out as the most accurate (2 According to Table 24, under low lighting, accuracy dropped significantly for all methods, especially for Azure (40 Another aspect considered for adjusted results was distance to the capture point. Table 25 shows that at adequate distances, AWS maintained the lowest error margin (21 On the other hand, under poor distance conditions, counting was severely compromised, representing high sensitivity to object

Table 23 – Performance for each cloud service provider under ideal lighting conditions calculated from the number of accumulated detections among all analyses and the expected number of detections.

Provider	Detections	Expected	Absolute Error	Percentage Error
Azure	44	48	4	8%
AWS	47	48	1	2%
GCP	44	48	4	8%
Edge	33	48	15	31%

Source: Elaborated by the author.

Table 24 – Performance for each cloud service provider under poor lighting conditions calculated from the number of accumulated detections among all analyses and the expected number of detections.

Provider	Detections	Expected	Absolute Error	Percentage Error
Azure	24	43	19	40%
AWS	33	43	10	21%
GCP	27	43	16	33%
Edge	22	43	21	44%

Source: Elaborated by the author.

Table 25 – Performance for each cloud service provider under ideal distance conditions calculated from the number of accumulated detections among all analyses and the expected number of detections.

Provider	Detections	Expected	Absolute Error	Percentage Error
Azure	35	29	8	17%
AWS	39	29	4	8%
GCP	35	29	8	17%
Edge	26	29	17	35%

Source: Elaborated by the author.

distance relative to capture. For example, all providers had equally poor behavior (83 These

Table 26 – Performance for each cloud service provider under poor distance conditions calculated from the number of accumulated detections among all analyses and the expected number of detections.

Provider	Detections	Expected	Absolute Error	Percentage Error
Azure	3	20	17	83%
AWS	3	20	17	83%
GCP	3	20	17	83%
Edge	1	20	19	88%

Source: Elaborated by the author.

results indicate 5 factors that impair vehicle tracking accuracy:

1. adverse conditions (rain and poor lighting - night);
2. recording angle (diagonal versus profile);
3. distance to capture point;

4. quality of object detection stage; and

5. tracking system parameterization (*deepsort*).

6.3 Processing performance

Another very important parameter in application evaluation is performance which can be evaluated from different metrics. In computer vision, a commonly used metric in application evaluation is FPS (*frames* or frames per second), which indicates exactly how many frames or images are processed in 1 second. Most cameras are capable of recording video at least at 30 FPS. Other cameras with more advanced technology can record at 60 and even 120 FPS. A higher number of frames per second contributes to better quality detection and counting results, since the displacement between 2 frames is smaller, allowing the tracking algorithm (*deepsort*) to better follow object movement and be less sensitive to the speed at which the object moves between frames. On the other hand, recordings with high FPS values also represent an increase in detection stage cost (since cost is directly proportional to the number of frames to be analyzed). Another important factor is that the increase in frame number may hinder real-time processing. If an application is capable of processing 15 FPS and capture occurs at 30 FPS, what happens is that every second that passes, the application will be delayed in processing 15 frames that will be added to a queue to be processed, making real-time processing impossible. For the analyzed videos, the application processing speed results in frames per second are exposed in Table 27, calculated based on complete processing - detection, tracking and counting. In general,

Table 27 – Processing performance results in FPS (frames per second) in object detection and tracking per video for each cloud service provider and for edge processing.

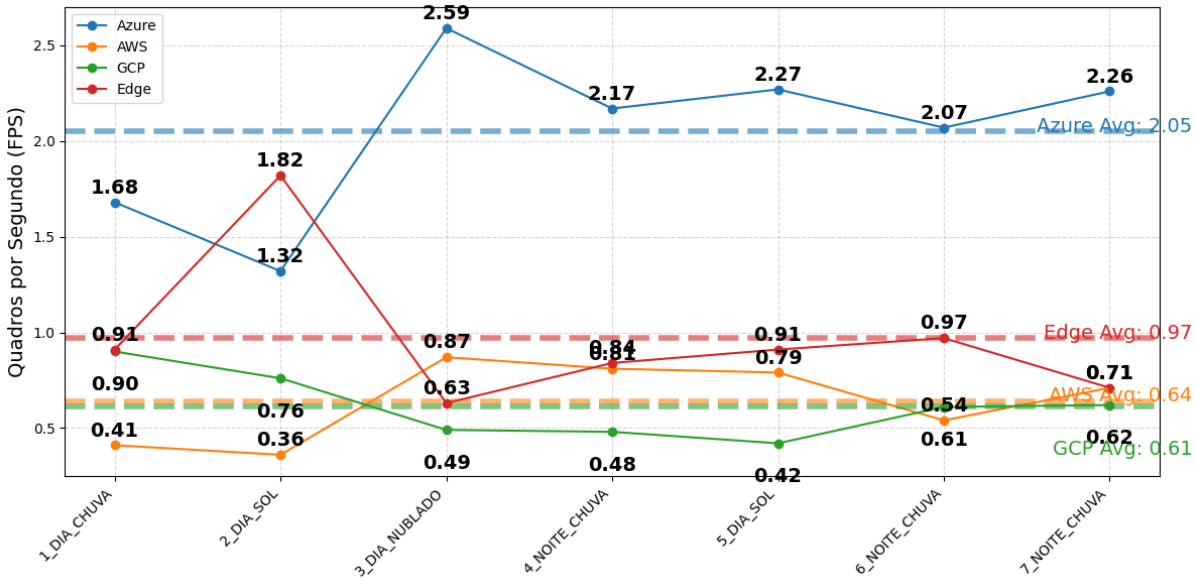
ID	Total frames	FPS Azure	FPS AWS	FPS GCP	FPS Edge
1	1849	1.68	0.41	0.90	0.91
2	3611	1.32	0.36	0.76	1.82
3	5416	2.59	0.87	0.49	0.63
4	5411	2.17	0.81	0.48	0.84
5	1827	2.27	0.79	0.42	0.91
6	3652	2.07	0.54	0.61	0.97
7	1816	2.26	0.71	0.62	0.71

Source: Elaborated by the author.

Azure environment presented the best average FPS performance, surpassing other providers in all tested videos. The average FPS obtained with Azure was 2.05, with peaks up to 2.59 FPS (3_DIA_NUBLADO). AWS presented the worst overall performance, with FPS varying between 0.36 and 0.87, and average of 0.64 FPS. This can be attributed to task initialization *overhead* and lower efficiency in resource provisioning for short loads typical of video applications with inference. Google Cloud Run (GCP) presented intermediate results, with FPS varying from 0.42

to 0.90, and average of 0.61 FPS. Despite performance similar to AWS in some videos, GCP showed greater variability between executions. Edge processing presented stable and competitive performance, with FPS varying between 0.63 and 1.82, and average of 0.97 FPS. Particularly, in video 2, performance was superior to all cloud platforms, including Azure. This result suggests that, for specific and well-optimized scenarios, using embedded devices can be a viable and efficient alternative, especially when there are connectivity restrictions or demand for reduced latency. This behavior can also be observed in graph Figure 31, which makes evident that Azure is capable of consistently processing more frames per second than the others. While AWS and GCP have opposite performances, but with very close averages. Edge processing demonstrating slightly better than AWS and GCP. This result is predominantly linked to the detection stage, for which Azure's response time (latency) is lower, enabling faster processing. During the tracking and counting stage, processing time between different providers is similar since they use the same *deepsort* implementation. Besides frames per second, another evaluated metric was response

Figure 31 – FPS comparison per cloud service provider and edge processing.



Source: Elaborated by the author.

latency, as presented in Table 28. This latency is calculated only over "detection" requests made to each provider's API and edge service, and are calculated as the time difference between the moment immediately before the request and the moment immediately after receiving the request response. Azure processing demonstrated superiority regarding latency, with average of 333 ms per request. Another provider that demonstrated quite interesting latency values was AWS, with average of 440 ms per request. In general, the higher the latency, the worse the service result since there is greater delay between sending a request and receiving the response. Higher latency also implies a higher time needed to process a complete video, becoming an impediment regarding real-time processing. From latency, the exclusive FPS average of the detection stage

Table 28 – Average latency performance results (ms) in object detection and tracking per video for each cloud service provider and for edge processing.

ID	Azure Latency	AWS Latency	GCP Latency	Edge Latency
1	322	540	1086	1375
2	369	466	1734	1358
3	314	396	856	1351
4	344	455	1031	1374
5	341	393	1039	1339
6	326	420	1120	1373
7	319	415	836	1349

Source: Elaborated by the author.

Note: In this table latency was calculated as the average between the latency of all analysis requests made for each video.

for each provider can be calculated indirectly, according to Table 29. This exclusive FPS of the detection stage indicates on average how many video frames or how many images per second it is possible to process using each provider. In this case, higher values represent better performance, since they indicate that the service can process more images or frames per second, favoring, for example, applications that have real-time processing as a requirement. Processing through

Table 29 – Average FPS calculated using average latency and representing an exclusive FPS result of the detection stage for each cloud service provider and for edge processing.

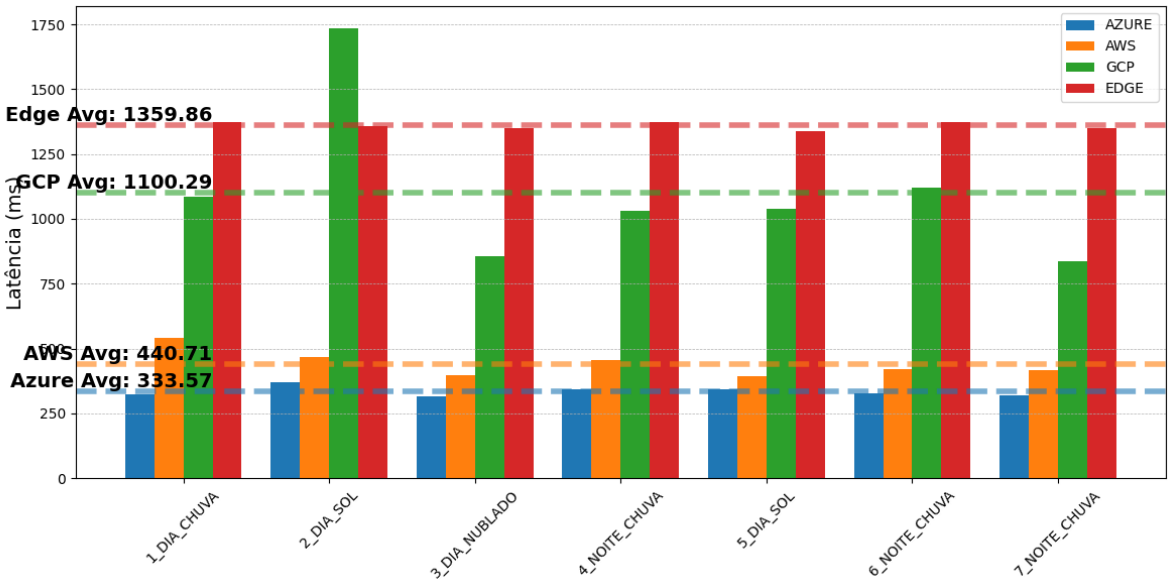
Processing Source	Average Latency	Average FPS (detection)
Azure Vision AI	333 ms	3.00
Amazon AWS Rekognition	440 ms	2.27
Google Cloud Vision	1100 ms	0.90
Edge (Raspberry Pi)	1359 ms	0.73

Source: Elaborated by the author.

Note: In this table the average latency was calculated as the average of each video's latencies, thus representing a higher level of aggregation.

Google API and edge service had substantially worse performance, with averages of 1100 ms and 1359 ms, respectively. Besides averages, the graph in Figure 32 shows that this behavior was consistent among all videos, even though processing of them happened at very different times from each other. The high latency of edge processing results from Raspberry Pi limitations that stem mainly from device hardware restrictions, which does not have GPU acceleration and has limited computational resources. On the other hand, among the 4 analyzed options (Azure, AWS, GCP and Raspberry Pi), the only option that can be customized and come to present better results is precisely edge processing. Replacing the Raspberry Pi with a device with more processing resources (such as an NVIDIA Jetson Nano) would result in considerable latency reduction, in exchange for higher capital investment. Besides latency, another important parameter is total processing time, which does include both tasks - detection and tracking. Table 30 presents

Figure 32 – Latency in object detection per cloud service provider and edge processing.



Source: Elaborated by the author.

total processing times (in seconds) for executing object detection and tracking tasks in videos with different frame quantities. These times directly reflect the computational capacity of each evaluated environment, considering the total workload imposed by each video. Again, Azure

Table 30 – Processing time (s) in object detection and tracking per video for each cloud service and for edge processing.

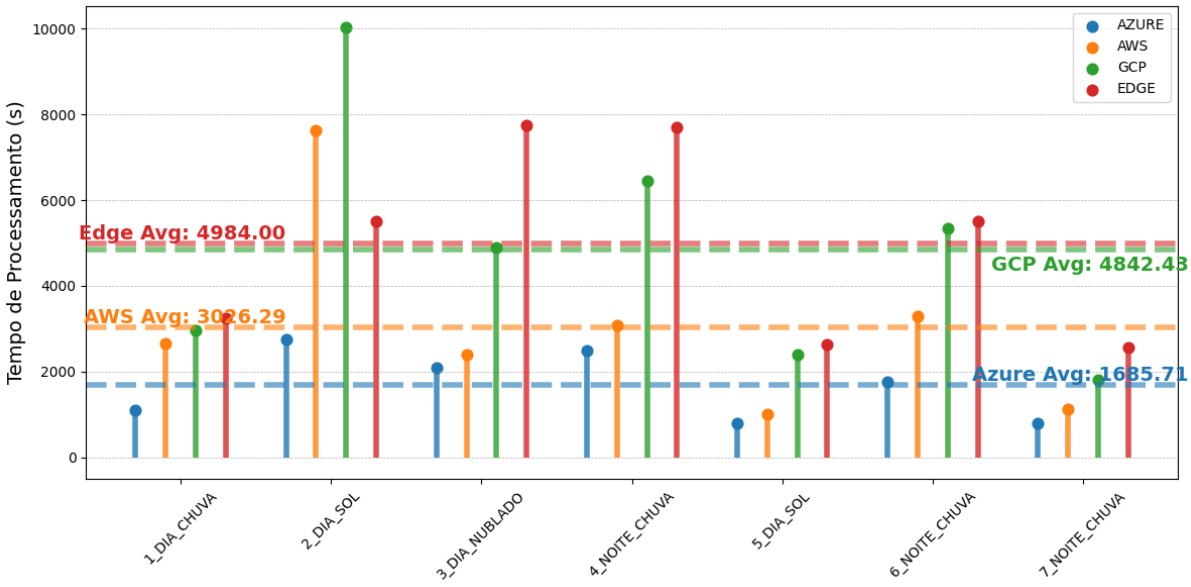
ID	Frames	Azure Time	AWS Time	GCP Time	Edge Time
1	1849	1102	2659	2959	3237
2	3611	2742	7621	10023	5501
3	5416	2088	2388	4897	7751
4	5411	2493	3091	6459	7689
5	1827	805	1015	2395	2636
6	3652	1766	3283	5351	5507
7	1816	804	1127	1813	2567

Source: Elaborated by the author.

processing presented, consistently, the lowest processing times in almost all videos, achieving an average of 1685 seconds. This reinforces the results observed in latency and FPS metrics, evidencing that the computer vision solution provided by Azure is capable of maintaining high performance with varied loads and at different periods of the day. AWS obtained intermediate results, with an average of 3026 seconds (significantly higher than Azure's average). In GCP's case, processing times were the highest among cloud providers in almost all videos, reaching over 10 thousand seconds in video 2_DIA_SOL. Edge processing, although presenting lower times than GCP in some cases (such as in video 2_DIA_SOL), maintained inferior performance

in shorter and medium videos. Edge processing is slower due to limited resources available for task execution. For GCP, processing time is also high due to high latency in the detection stage. Such results are also demonstrated graphically in Figure 33.

Figure 33 – Processing time per video for different cloud service providers and for edge computing.



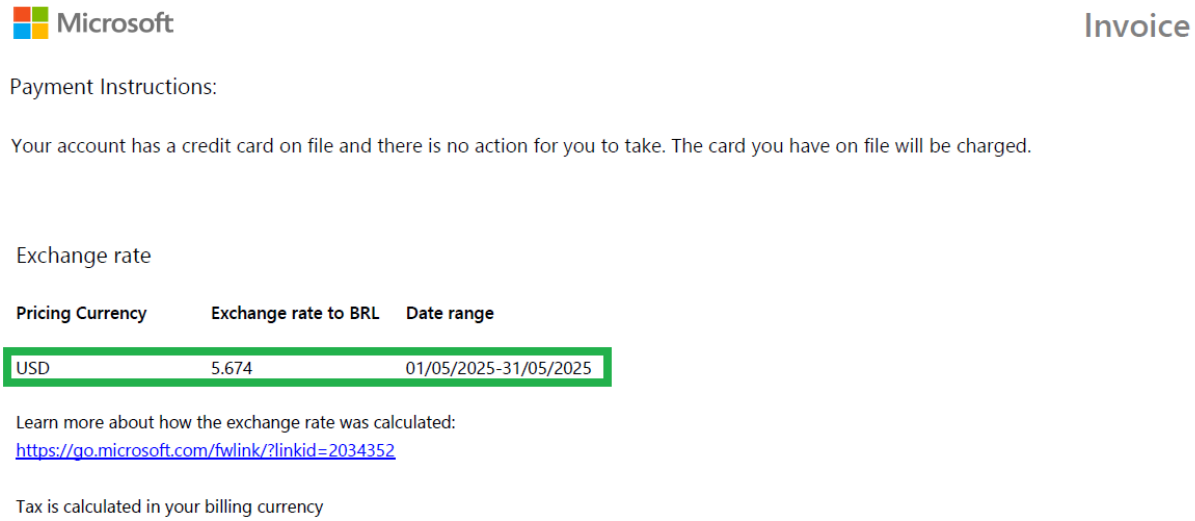
Source: Elaborated by the author.

6.4 Total development costs and hidden cost impacts

For technology viability it is important to analyze costs as a whole, expanding the scope from the initial one that was limited exclusively to processing costs of each video, to also encompass costs associated with solution development as a whole. The results presented so far described exclusively the costs (and results) associated exclusively to processing each video, but do not describe the totality of costs associated with solution development, so a more detailed cost analysis is appropriate. This cost analysis in cloud service environments refers to systematic evaluation of computational resource expenses and is fundamental for financial planning, to optimize resource use and to reduce waste. Cost analysis also allows identifying usage patterns that justify strategic changes, such as service migration, infrastructure reconfiguration or adoption of hybrid solutions. A common problem associated with cloud service costs is *hidden costs*, which stem from lack of clarity in documentation, ambiguity in pricing models, billing for implicit resources (such as internal transfers, indirect API calls or service subcomponents), or even inadvertent adoption of configuration patterns that maximize consumption. These situations not foreseen by development and business teams compromise the planned budget and can make continuous tool use unfeasible. Also, if not considered as part of the final cost, they can result in incorrect service pricing and deficit financial operation (PETCHIAPPAN, 2024). For each

provider, USD to BRL exchange rates were considered according to invoices or fiscal notes obtained as service billing (Figure 34, Figure 35 and Figure 36). Table 31 presents a consolidation

Figure 34 – Azure service invoice page indicating the exchange rate from USD to BRL.



Source: Elaborated by the author.

of costs related to main services used in Microsoft Azure provider for developing this solution during the research period. Values were organized in reais (BRL) and dollars (USD) (exchange rate of R$ 5.674 per dollar). Associated tax values are also described considering the 12.15% tax burden. It is observed that most of the cost was concentrated in cognitive services for image analysis. As discussed, the object detection and tracking solution focused on people and vehicles

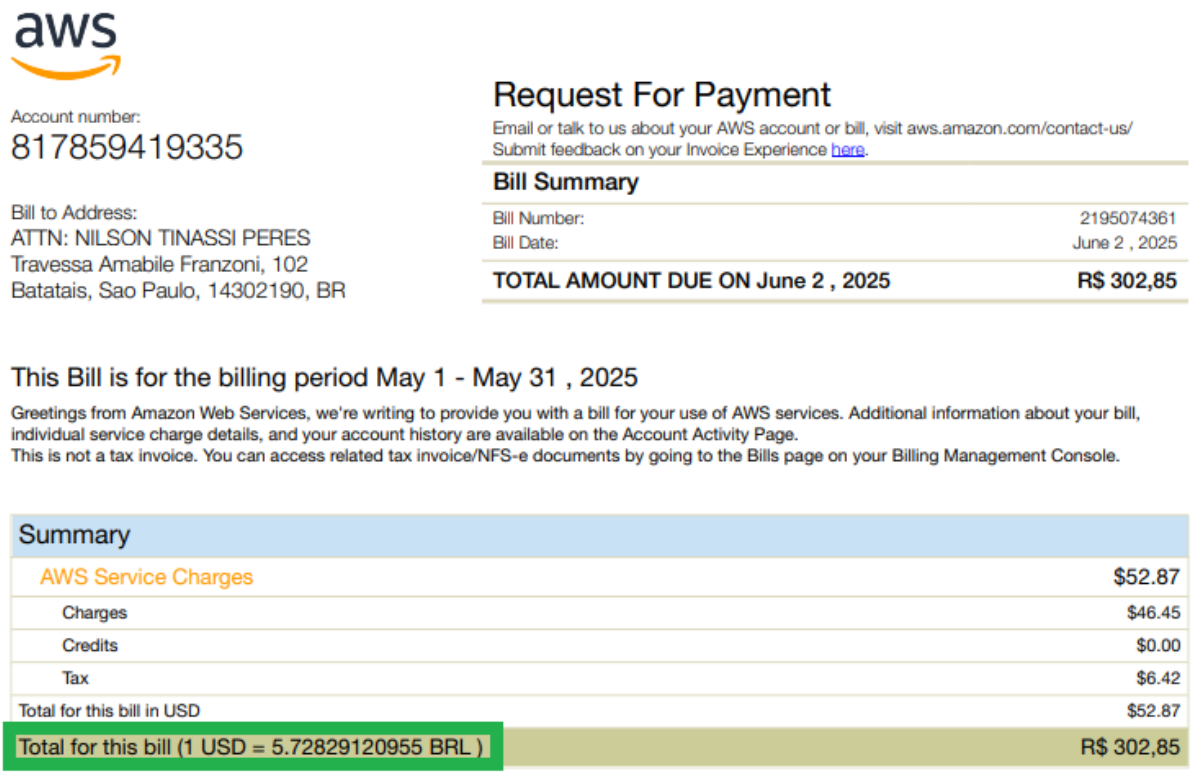
Table 31 – Total development costs by resource type and with tax addition - Azure.

Resource Type	**Description**	**Cost**	**Taxes**	**Total**
Cognitive Services	Image API - Objects	R$ 350.33 (U$ 61.62)	R$ 48.36	R$ 398.69
Cognitive Services	Image API - People	R$ 350.33 (U$ 61.62)	R$ 48.36	R$ 398.69
Log Analytics	Workspace	R$ 84.45 (U$ 14.88)	R$ 11.65	R$ 96.10
Container App	Execution	R$ 39.08 (U$ 6.89)	R$ 5.39	R$ 44.47
Grand Total		**R$ 824.19**	**R$ 113.76**	**R$ 937.95**

Source: Elaborated by the author.

uses *Azure Computer Vision API*. The main call of this API was configured to simultaneously use two sub-analyses: VisualFeatures.OBJECTS and VisualFeatures.PEOPLE. However, Microsoft's official documentation does not directly explain that each sub-analysis is billed as an individual transaction. Nor does it clarify that the OBJECTS sub-analysis already includes people detection, albeit with lower precision. This behavior led to cost duplication for each processed image. When using both visual features together, billing was performed separately for each one, resulting in two transactions per image. Such configuration, although technically valid and useful in terms of result quality, caused a significant elevation in total cost. This situation is a classic example

Figure 35 – AWS service invoice page indicating the exchange rate from USD to BRL.



Source: Elaborated by the author.

of *hidden cost*, widely reported in critical analyses about cloud solution use (PETCHIAPPAN, 2024). Table 32 presents costs resulting from using Amazon Web Services (AWS) infrastructure

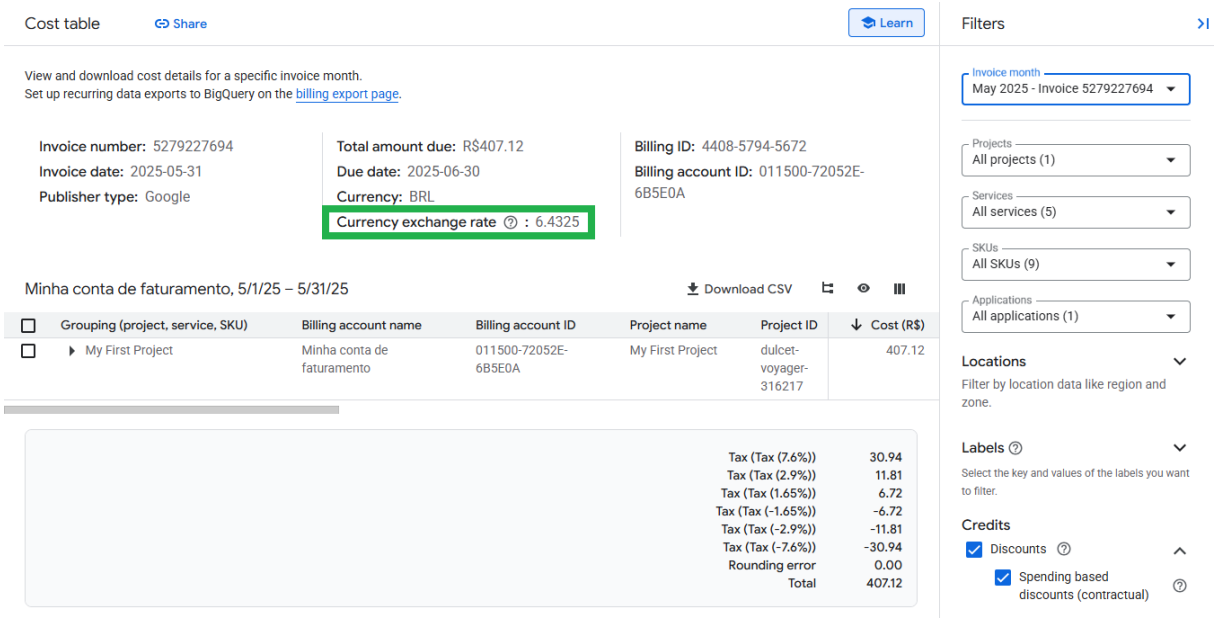
Table 32 – Total development costs by resource type and with tax addition - AWS.

Resource Type	**Description**	**Cost**	**Taxes**	**Total**
Fargate	Task execution	R$ 290.19 (U$ 50.63)	R$ 40.04	R$ 330.23
Rekognition	DetectLabels API	R$ 161.68 (U$ 28.22)	R$ 22.31	R$ 183.99
Rekognition	Processed images	R$ 134.65 (U$ 23.50)	R$ 18.58	R$ 153.23
EC2	No operation (idle)	R$ 92.30 (U$ 16.11)	R$ 12.74	R$ 105.04
Cloud Run	Service execution	R$ 50.60 (U$ 8.84)	R$ 6.98	R$ 57.58
EC2	DescribeNetwork	R$ 29.38 (U$ 5.13)	R$ 4.06	R$ 33.44
Grand Total		**R$ 758.80**	**R$ 104.71**	**R$ 863.51**

Source: Elaborated by the author.

during solution development. Original values are in dollars and were converted to reais using the exchange rate of R$ 5.73 per dollar, with addition of 12.15% taxes. It is observed that the highest costs are associated with task execution in Fargate and use of Rekognition service API. The cost analysis here reveals another relevant example of unexpected cost, related to AWS Fargate use (includes Fargate and EC2 from the table). Technical documentation allowed the conclusion that billing was performed based on effective task execution time (*on-demand*). However, in

Figure 36 – GCP service invoice page indicating the exchange rate from USD to BRL.



Source: Elaborated by the author.

practice billing encompasses all infrastructure provisioning time and all allocated environment (including unused resources), regardless of actual processing execution. A more careful study of documentation performed later revealed that this pricing strategy was documented, but with certain ambiguity that could lead to erroneous interpretations. This scenario generated significant, unnecessary and unexpected expense. After identifying this financial bottleneck, the architecture was modified to adopt AWS Cloud Run service, which, besides being more adherent to *serverless* model, resulted in significantly lower costs. Although the unexpected cost was consequence of imprecise documentation interpretation, it is a recurring risk in cloud environments, especially when billing models are not transparent or demand specialized technical reading. Table 33

Table 33 – Total development costs by resource type and with tax addition - GCP.

Resource Type	**Description**	**Cost**	**Taxes**	**Total**
Vision AI	Cloud Vision API	R$ 399.57 (U$ 62.12)	R$ 55.26	R$ 454.84
Cloud Run	Service execution	R$ 0.00 (U$ 0.00)	R$ 0.00	R$ 0.00
Grand Total		**R$ 399.57**	**R$ 55.26**	**R$ 454.84**

Source: Elaborated by the author.

presents costs associated with using Google Cloud Platform (GCP) in solution development. The only recorded cost was Cloud Vision API use, with value of R$ 454.84 (exchange rate of R$ 6.4325 per dollar), which includes 12.15% taxes. Cloud Run service costs were null, since usage stayed within the monthly free quota provided by the platform: 180,000 vCPU-seconds and 360,000 GiB-seconds. Unlike cases observed in AWS and Azure, there was no occurrence of unexpected costs in GCP service configuration and use. This behavior is attributed to the

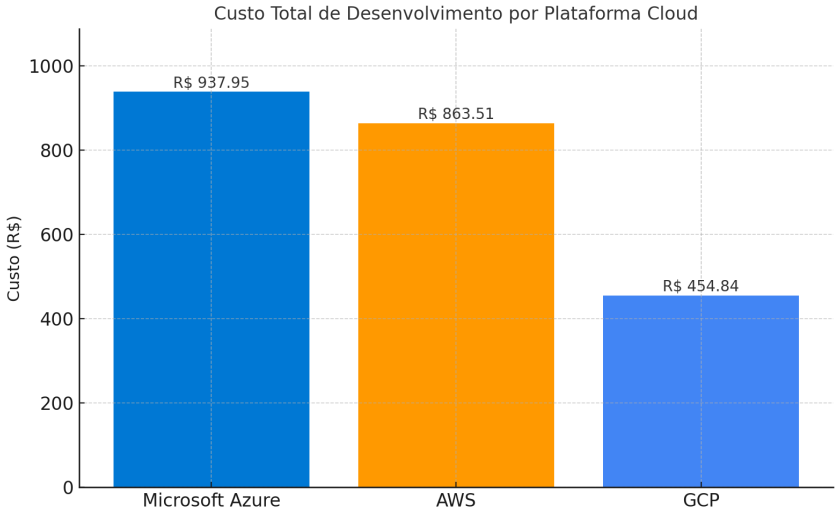
fact that GCP was the last environment to be configured and that development was performed with greater caution, thoroughly exploring technical documentation and service pricing policies before implementation (lessons learned from Azure and Amazon AWS errors and surprises). Despite taxes applied to services from all three platforms — Azure, AWS and Google Cloud — being the same, totaling 12.15% (comprising PIS, COFINS and ISS), a significant disparity is observed in dollar to real conversion rates used by each provider. In Azure's case, a rate of R$ 5.674 per dollar was applied, while AWS used R$ 5.73 per dollar. Google Cloud adopted the highest rate, R$ 6.4325 per dollar. These variations directly influence the final cost of services in reais, regardless of the original value in dollars, and should be carefully considered in any comparative cloud cost analysis. The cost summary evidences that Microsoft Azure was the

Table 34 – Summary of total development costs by provider

Platform	**Total Cost (R$)**
Microsoft Azure	R$ 937.95
Amazon Web Services (AWS)	R$ 863.51
Google Cloud Platform (GCP)	R$ 454.84
Grand Total	**R$ 2,256.30**

Source: Elaborated by the author.

Figure 37 – Total development cost by cloud service provider



Source: Elaborated by the author.

platform with highest cost in the application development process (R$ 937.95), followed by Amazon Web Services (R$ 863.51), while Google Cloud Platform presented the lowest cost (R$ 454.84) (according to Table 34 and Figure 37). This difference is directly related to how each provider was used during the project. Azure was the initial base for solution development, having been employed in testing phases of versioning and optimization. Therefore, it was the most used service, naturally accumulating higher cost. AWS, in turn, was the second most used

platform, largely due to significant rework, such as the need to migrate from AWS Fargate — which generated unexpected costs — to AWS App Run. GCP was the last to be implemented, benefiting from maturity acquired throughout development. With a more cautious approach based on previous learnings, GCP implementation required less testing and adjustments, resulting in significantly lower final cost.

CHAPTER

7

CONCLUSÃO

This section presents the main final reflections of this work based on the results obtained throughout the conducted experiments. From the comparative evaluation between different architectures (cloud and embedded), service providers (Azure, AWS, and GCP), and artificial intelligence approaches applied to the problem of people and vehicle detection and counting, relevant lessons were extracted about the impacts of technological decisions in real computer vision projects.

More than pointing out which solution achieved better performance in a given aspect, this conclusion seeks to consolidate a critical and strategic view of the trade-offs involved — such as cost versus quality and performance versus implementation complexity. The learnings synthesized here serve as practical guidelines for professionals and researchers who face similar dilemmas when choosing between different technological solutions for specific problems.

Thus, the initial conclusions concern architectures for real-time image analysis. The solutions offered by cloud computing providers (Azure, AWS, or GCP) are not capable of natively meeting solutions that require real-time image processing using the image detection API. This conclusion is based on the FPS results for each provider considering only the object detection task, which according to Table 29, were Azure: 3 FPS, AWS: 2.27 FPS, and GCP: 0.90 FPS. These values are significantly below the 30 FPS considered as the minimum reference for applications requiring real-time video processing.

This conclusion is reinforced when considering the complete analysis pipeline that includes object detection and tracking, for which the results of Figure 31 were obtained: Azure: 2.05 FPS, AWS: 0.64 FPS, and GCP: 0.61 FPS.

These results refer exclusively to the object detection APIs provided by each of the providers. An alternative suggested by cloud computing providers for real-time video analysis is the use of video streaming processing. This alternative achieves higher FPS rates, but besides also not reaching the 30 FPS baseline for real-time processing considered as an architecture

requirement, it also has other limitations. One of these limitations is that object detection processing using streaming returns at the end of the analysis a dataset with a list of objects per video period, and not per frame.

This type of result adds some overhead to analyses with image pipelines, since it is necessary to create a mapping between each second of the video and the frames they refer to, to then select the frames individually in the subset to send to the next processing stages (in the example considered by this work, the next processing stage would be performing tracking, unique identification, and object counting). Another problem with using streaming processing is the pricing that follows a unique structure apart from image analysis pricing. For example, object detection via AWS Rekognition streaming costs $0.00817 per minute, while detection per frame costs $0.0010 per image (or frame). In this scenario, there is an advantage in streaming analysis as long as this analysis can perform processing of at least 9 images (or frames) per second (FPS).

In the cloud computing environment, it is important to consider the processing time, since a hypothesis could affirm that during peak hours the request response time is affected, but this is not the case since regardless of the request time the results always had the same behavior, Azure with the lowest latency, then AWS, then GCP, and finally edge processing. The only exception is for day 2_DIA_SOL, in which Google's latency reached levels of almost 1750 ms, which is 50% more than the average latency of this provider.

It should also be considered that besides the ready-made image analysis solutions made available by cloud computing providers, there is alternatively the possibility of developing custom solutions. In this case, the developer is responsible for architecture design, tool selection, and implementation. This strategy offers greater flexibility and is the most recommended for real-time image analysis, since it allows, among other factors, selecting the resources that will be available (CPU, GPU, and memory). With more available resources, it is possible to perform processing tasks in less time and leverage the amount of FPS to achieve real-time processing.

A viable alternative to enable real-time image processing using cloud APIs is the adoption of architectures based on multiprocessing. Instead of making sequential calls to the object detection API, this approach executes multiple simultaneous calls, promoting parallelism in processing. Although the individual latency of each request remains similar, the fact that responses are obtained in parallel prevents the accumulation of these times, which allows meeting the real-time requirement, provided that the volume of calls and parallelism capacity are adequately dimensioned.

This approach, however, also has its limitations. For example, in the example analyzed by this work, it is mandatory that the object tracking task be performed in each video frame sequentially (since the DeepSORT algorithm tracks the movement trajectory of objects). This means that in some pipelines only some tasks could be optimized using multiprocessing, but still with relevant gains.

In summary, the main cloud providers do offer robust solutions for image analysis, but their native detection APIs do not meet the minimum performance requirements for real-time applications. The suggested alternatives, such as the use of streaming or architectures based on multiprocessing, offer some relief, but introduce new challenges — whether in implementation complexity or in technical limitations, such as the impossibility of parallelizing sequential tasks like object tracking. Moreover, the cost-benefit of these alternatives depends directly on the FPS effectively achieved, which requires care in usage estimation. Thus, it is concluded that only custom solutions, with total control over computational resources and pipeline logic, are truly capable of achieving the 30 FPS required by critical real-time applications, so that the choice of the most adequate architecture must consider the complete application pipeline, its bottlenecks, and the relationship between performance, cost, and development complexity.

From a performance perspective, the experiments with edge processing led to conclusions similar to those obtained with the use of cloud APIs. The processing rate achieved was approximately 0.73 FPS in the exclusive detection stage, a value much lower than the 30 FPS required for real-time applications. This limited performance is directly related to the constraints of the hardware used — a Raspberry Pi 4 with 4 GB of RAM, low-performance CPU, and absence of dedicated GPU. This limitation highlights the importance of careful choice of the embedded device: solutions based on edge processing are only viable for real-time scenarios when they have more robust devices, such as NVIDIA Jetson, which offer hardware acceleration. Other advantages of edge processing include privacy, independence from internet connection, latency predictability, continuous operational cost, and capacity for solution customization — characteristics that can be determining factors in specific applications.

Still in this sense, the choice to use the Yolo v11 nano model with the objective of compensating for the limited resource availability of the Raspberry Pi was only partially successful. This conclusion is reached because it was not possible to achieve a good processing rate even using a theoretically lighter model adapted for embedded processing and because the quality of the results ended up still being compromised by the use of this lighter model (note that the quality results were worse for edge processing than for cloud computing providers, since there is a loss of precision in exchange for a lighter model). However, for more comprehensive conclusions, a comparison between models would be necessary, such as Yolo v11 nano and Yolo v11x, which is beyond the scope of this work.

Regarding the quality of results, Azure obtained better results for people counting (Figure 23 - MAPE[1] of 41.7%), while AWS obtained better results for vehicle counting in ideal lighting conditions (MAPE of 2%), in poor lighting conditions (MAPE of 21%), in ideal distance conditions (MAPE of 8%), and in poor distance conditions (MAPE of 83%, tied with Azure and GCP), respectively according to tables: Table 23, Table 24, Table 25, and Table 26.

These results indicate that the choice of provider depends on the type of object to be

[1] Mean Absolute Percentage Error.

detected and operational conditions. The experiments conducted with Azure, for example, used an API optimized for people analysis, contributing directly to its better result in this category. It should be noted, however, that the test samples with people were small, so for additional conclusions it is important to conduct additional tests with more comprehensive samples.

Still considering the importance of selecting the most appropriate provider, AWS presented superior and more consistent performance in vehicle counting, especially in controlled conditions. This suggests that, for specific applications — such as traffic monitoring or vehicular access control — AWS tends to be more adequate.

Also, the high percentage error in low luminosity and large distance scenarios evidences a common limitation among providers (including edge processing), indicating that adjustments in image capture (such as camera positioning or use of auxiliary sensors) are essential to improve solution quality and also indicating that operational conditions are determining factors in project architecture design and associated limitations.

The tuning or parameterization of the deepsort model used is one of the configurations that allow improving solution quality. Despite being unfeasible to perform parameterization for each video (since it would go against the tool's proposal to allow analysis of any input video sent by the user), a provider-level parameterization was performed. The results obtained for video 1_DIA_CHUVA (used as the parameterization base) were significantly better for all providers, indicating that it is possible to compensate for variables related to operating conditions. On the other hand, the results for the other videos exemplify the difficulty of generalizing parameters to obtain a consistent response for any input video, thus suggesting that the ideal architecture should be dynamic to self-adapt to different scenarios.

Also, the discrepancies in counting results between providers, as in the case of video 1_DIA_CHUVA, which presented 22 objects counted in GCP against only 13 in Azure, are attributed mainly to the object detection stage, which precedes tracking. The conclusion is that even though the tracking and counting logic is standardized among different providers, the accuracy of final counting depends directly on the quantity and quality of detections performed by each service.

Specifically about the algorithms used by each cloud computing provider, some limitations were identified, such as identifying groups of people as a single person or identifying the driver of a motorcycle or car as an object separate from the vehicle itself. This second case is quite particular since it could be considered expected behavior or not depending on the project requirements. For some projects, detecting the vehicle and the person can be considered correct behavior, so this type of limitation should be treated via post-processing of results.

Cost analysis was one of the main objectives of this work and revealed that, in general, GCP was the most costly platform for performing object detection in images, while Azure and AWS have similar prices. Among the alternatives tested, edge processing was by far the lowest

cost option.

It should be noted that the pricing structure between different providers is similar, with prices associated with API calls or analyzed images and with progressive discounts based on the quantity of events in a given period. Due to this structure, image analysis prices using cloud computing become similar even through different providers, which contributes to a greater weight of other factors in the decision about which provider to use in designing a solution architecture.

A common hypothesis is that edge processing cost can exceed cloud processing cost mainly in the short term due to necessary investments with hardware acquisition for embedded processing and environment configuration. This hypothesis was confirmed by this work, in which an investment of R$ 729.00 (Raspberry Pi 4 and accessories) was necessary to start development, while no initial investment was necessary to start development in the cloud environment (only user account creation).

However, this hypothesis does not adequately represent all costs involved during the prototyping and development phase of a solution. During this phase, there is a need to execute and validate the solution already using cloud computing resources before even having the finished product. Thus, although edge processing required an initial investment of R$ 729.00, this was the only necessary cost throughout the entire development process. In contrast, despite cloud-based solutions not requiring initial investment, they present significant accumulated costs during the prototyping and validation phase. In this study, the total spent on cloud computing during the prototyping and development stage was R$ 2,256.30.

The analysis also evidenced an important difference in the cost profile between approaches: the cloud model dilutes investment over time (pay-as-you-go or pay-per-service), but imposes continuous operational cost even before the solution is in production. Edge processing concentrates costs at the beginning, but tends to be more economical in scenarios with high volume of tests and experiments. Therefore, in projects with long development and validation phases, or with restricted budget for recurring costs, edge may represent a more financially advantageous alternative.

Thus, despite the initial investment in hardware and eventual scalability or performance limitations, the work results evidence the economic viability of local processing for continuous use scenarios, prototyping and validation, or large-scale operation.

However, it is important to highlight that edge processing may not be adequate for all contexts, especially when there is need for rapid scalability, high availability, or integration with other cloud-based services. In these cases, the higher cost of cloud may be justified by the robustness of the offered infrastructure.

Cost evaluation also revealed the lack of clarity in cloud computing solution specifications that often have hidden costs and that make comparability between providers difficult as well as cost planning, whether short, medium, or long term. For example, although Azure and AWS

suggest the same price for each object detection event ($0.0010 per event), the US dollar to Brazilian real exchange rates are different: Azure - U$ 1.00 = R$ 5.674; Amazon: U$ 1.00 = R$ 5.730). Despite the difference between values being small, it can become relevant in large-scale scenarios. Comparing any of these two providers with the exchange rate billed by Google (U$ 1.00 = R$ 6.4325), a substantially larger difference is found, reinforcing the need to carefully evaluate the costs involved.

Other hidden costs associated with lack of clarity occur due to misuse or misconfiguration of available solutions, such as using AWS Fargate instead of AWS App Run or consuming 2 events per Azure image analysis API call instead of just 1 as expected[2].

A conclusion resulting from these errors is that knowledge of services is vital for adequate solution development, so that investment in quality workforce with high technical knowledge of the cloud environment is very important to avoid being surprised by unexpected costs.

Besides cost, quality, and performance aspects, there are other factors that influence the determination of a computer vision system architecture, such as the type of file to be processed (image, video, or streaming), file size, and even file format. These factors are important due to the limitations of each cloud computing provider. For example, image analysis by Azure may have better performance, but may not support exactly the same file formats that are supported by AWS. These situations may force the use of a solution that is not optimized for that scenario, but that is the only one that satisfies the problem requirements.

The final conclusions are: it is not possible to achieve real-time processing using the suggested architecture, but it is possible to perform processing asynchronously, making results available after operation completion; Azure is the cloud computing platform with lowest cost for vehicle and people detection and AWS is the platform with best results for vehicle and people tracking; edge processing is an economically viable alternative, but with loss of quality and performance, these can be compensated by using hardware with greater resource availability.

This work provides relevant contributions to the area, such as: the discussion and comparison of computer vision architecture design using cloud computing and edge computing; cost analysis and comparison between different providers and for edge processing; detailed evaluation of the performance of the main commercial object detection APIs in real scenarios; identification of the main bottlenecks and technical limitations of each approach, especially in the context of real-time applications; and the proposition of alternatives and strategies, such as the use of

[2] This additional cost could have been partially avoided. Using only the OBJECTS sub-analysis would allow identification of people and vehicles at a lower cost, since it would generate only one transaction per image. However, this approach would sacrifice the precision and level of detail offered by the PEOPLE sub-analysis, which would compromise detection quality in more complex scenarios. Considering the application objectives — which require high accuracy in people identification in varied environments — it was decided to maintain both sub-analyses, even with doubled cost. This decision is in line with good solution engineering practices, in which result reliability justifies a higher financial investment.

multiprocessing and pipeline customization, to mitigate performance and cost restrictions. Additionally, the work highlights the importance of careful hardware choice in embedded solutions, the need for deep technical knowledge to avoid hidden costs in cloud environments, and offers practical recommendations for architecture selection according to specific requirements of each application.

The results of this work can be extended in the future by: increasing the scope of people and vehicles and repeating the analyses; including comparisons between different models both for detection, such as Yolo v11n and Yolo v11x, and for tracking, such as ByteTrack and StrongSORT; implementing the same architecture, but using hardware with more resources (NVIDIA Jetson or local server optimized for image processing); improving post-processing to filter detections that do not correspond to use case requirements (for example, considering person on motorcycle as only motorcycle detection, instead of a motorcycle and a person) or even by improving tracking by creating dynamic parameterization based on operating conditions such as luminosity, object distance, capture angle, and object speed.

In summary, this work evidences that, although cloud solutions offer practicality and scalability, they still do not fully meet performance requirements for real-time computer vision applications, especially regarding people and vehicle detection and counting. Edge processing, in turn, presents economic and privacy advantages, but depends heavily on the robustness of the hardware used. Thus, the choice of ideal architecture must carefully consider the balance between cost, performance, result quality, and the specific needs of each application. The analyses and comparisons presented here contribute to more informed decisions in future projects, serving as reference for both researchers and professionals who face similar challenges in developing computer vision systems.

BIBLIOGRAPHY

ABDULHAQ, K.; AHMED, A. A. Real-time object detection and recognition in embedded systems using open-source computer vision frameworks. **International Journal of Electrical Engineering and Sustainability (IJEES)**, College of Electronic Technology - Bani Walid, Bani Walid, Libya, v. 3, n. 1, p. 103–118, March 2025. ISSN 2959-9229. ISI Impact Factor 2023-2024: 0.557, Arab Impact Factor: 1.51, SJIF 2024: 5.274. Available: <https://ijees.org/index.php/ijees/index>. Citation on page 25.

AMAZON. **Amazon Rekognition Pricing**. 2025. Accessed: 2025-04-01. Available: <https://aws.amazon.com/rekognition/pricing>. Citations on pages 61, 62, and 78.

____. **What is AWS App Runner?** 2025. Acessado em: mai. 2025. Available: <https://docs.aws.amazon.com/apprunner/latest/dg/what-is-apprunner.html>. Citation on page 71.

____. **What is Cloud Computing?** 2025. Available: <https://aws.amazon.com/what-is-cloud-computing/>. Accessed: 27/03/2025. Citations on pages 52, 56, and 59.

____. **What is Computer Vision?** 2025. Available: <https://aws.amazon.com/what-is/computer-vision/>. Accessed: 22/03/2025. Citation on page 30.

Amazon Web Services (AWS). **What Is Amazon Rekognition Custom Labels?** 2025. <https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/what-is.html>. Acessado em 5 de julho de 2025. Citation on page 60.

____. **What is Amazon Rekognition? Image and Video Analysis APIs**. 2025. <https://docs.aws.amazon.com/rekognition/latest/dg/what-is.html>. Acessado em 5 de julho de 2025. Citations on pages 59 and 77.

ARMBRUST, M.; FOX, A.; GRIFFITH, R.; JOSEPH, A.; KATZ, R.; KONWINSKI, A.; LEE, G.; PATTERSON, D.; RABKIN, A.; STOICA, I.; ZAHARIA, M. Above the clouds: A berkeley view of cloud computing. **Electrical Engineering and Computer Sciences University of California at Berkeley**, 01 2009. Citation on page 55.

AZURE. **Azure Container Apps overview**. 2025. Acessado em: mai. 2025. Available: <https://learn.microsoft.com/en-us/azure/container-apps/overview>. Citation on page 71.

AZURE, M. **Responsible AI investments and safeguards for facial recognition**. 2022. Accessed: 2025-04-01. Available: <https://azure.microsoft.com/en-us/blog/responsible-ai-investments-and-safeguards-for-facial-recognition/>. Citation on page 63.

____. **Azure Computer Vision Pricing**. 2025. Accessed: 2025-04-01. Available: <https://azure.microsoft.com/en-us/pricing/details/cognitive-services/computer-vision/>. Citation on page 65.

____. **Azure Video Indexer Pricing**. 2025. Accessed: 2025-04-01. Available: <https://azure.microsoft.com/en-us/pricing/details/video-indexer/>. Citation on page 66.

BASTIAN, B. T.; C.V., J. Pedestrian detection using first- and second-order aggregate channel features. **International Journal of Multimedia Information Retrieval**, v. 8, n. 2, p. 127–133, 2019. ISSN 2192-662X. Available: <https://doi.org/10.1007/s13735-019-00171-0>. Citation on page 42.

BEWLEY, A.; GE, Z.; OTT, L.; RAMOS, F.; UPCROFT, B. Simple online and realtime tracking. In: **2016 IEEE International Conference on Image Processing (ICIP)**. [s.n.], 2016. p. 3464–3468. Available: <https://ieeexplore.ieee.org/document/7533003>. Citation on page 49.

BHUJEL, A.; WANG, Y.; LU, Y.; MORRIS, D.; DANGOL, M. A systematic survey of public computer vision datasets for precision livestock farming. **Computers and Electronics in Agriculture**, v. 229, p. 109718, 2025. ISSN 0168-1699. Available: <https://www.sciencedirect.com/science/article/pii/S0168169924011098>. Citation on page 32.

BONILLA, C.; HERMS, J.; MEDINA, O.; PEINADO, E. Discrete dark matter mechanism as the source of neutrino mass scales. **Journal of High Energy Physics**, Springer Science and Business Media LLC, v. 2023, n. 6, Jun. 2023. ISSN 1029-8479. Available: <http://dx.doi.org/10.1007/JHEP06(2023)078>. Citation on page 46.

BUYYA, R.; YEO, C. S.; VENUGOPAL, S.; BROBERG, J.; BRANDIC, I. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. **Future Generation Computer Systems**, v. 25, n. 6, p. 599–616, 2009. ISSN 0167-739X. Available: <https://www.sciencedirect.com/science/article/pii/S0167739X08001957>. Citation on page 55.

CANNON, S. E.; COLE, R. A. How accurate are commercial real estate appraisals? evidence from 25 years of ncreif sales data. **The Journal of Portfolio Management**, v. 37, n. 5, p. 68–88, 2011. Special Real Estate Issue. Citation on page 26.

CENTER, H. R. **The First Photograph**. 2025. Available: <https://www.hrc.utexas.edu/niepce-heliograph//>. Accessed: 22/03/2025. Citation on page 29.

CIAPARRONE, G.; SáNCHEZ, F. L.; TABIK, S.; TROIANO, L.; TAGLIAFERRI, R.; VAQUERO, A. L. M.; HERRERA, F. Deep learning in video multi-object tracking: A survey. **Neurocomputing**, v. 381, p. 61–88, 2020. Available: <https://www.sciencedirect.com/science/article/pii/S0925231219315586>. Citation on page 50.

CLOUD, G. **Google Cloud Vision Pricing**. 2025. Accessed: 2025-04-03. Available: <https://cloud.google.com/vision/pricing>. Citations on pages 69, 70, and 77.

DONG, X.; CAPPUCCIO, M. L. **Applications of Computer Vision in Autonomous Vehicles: Methods, Challenges and Future Directions**. 2024. Available: <https://arxiv.org/abs/2311.09093>. Citation on page 36.

DOSOVITSKIY, A.; BEYER, L.; KOLESNIKOV, A.; WEISSENBORN, D.; ZHAI, X.; UNTERTHINER, T.; DEHGHANI, M.; MINDERER, M.; HEIGOLD, G.; GELLY, S.; USZKOREIT, J.; HOULSBY, N. **An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale**. 2021. Available: <https://arxiv.org/abs/2010.11929>. Citation on page 40.

FOUNDATION, R. P. **Raspberry Pi 4 Model B specifications**. 2024. Acessado em: abr. 2025. Available: <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/>. Citations on pages 25 and 90.

GCP. **What is Cloud Run**. 2025. <https://cloud.google.com/run/docs/overview/what-is-cloud-run>. Acessado em: mai. 2025. Citation on page 72.

GERNSHEIM, H.; GERNSHEIM, A. **The History of Photography: From the Camera Obscura to the Beginning of the Modern Era**. [S.l.]: Oxford University Press, 1955. Citation on page 29.

GONG, Y.; LIU, G.; XUE, Y.; LI, R.; MENG, L. A survey on dataset quality in machine learning. **Information and Software Technology**, v. 162, p. 107268, 2023. ISSN 0950-5849. Available: <https://www.sciencedirect.com/science/article/pii/S0950584923001222>. Citation on page 32.

GOOGLE. **What is Cloud Computing?** 2025. Available: <https://cloud.google.com/learn/what-is-cloud-computing?hl=en>. Accessed: 27/03/2025. Citation on page 52.

Google Research. **Object Detection Model Card**. 2025. Accessed: 2025-04-02. Available: <https://modelcards.withgoogle.com/object-detection#performance>. Citation on page 68.

GUBBI, J.; BUYYA, R.; MARUSIC, S.; PALANISWAMI, M. Internet of things (iot): A vision, architectural elements, and future directions. **Future Generation Computer Systems**, v. 29, n. 7, p. 1645–1660, 2013. ISSN 0167-739X. Including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services & Cloud Computing and Scientific Applications — Big Data, Scalable Analytics, and Beyond. Available: <https://www.sciencedirect.com/science/article/pii/S0167739X13000241>. Citation on page 74.

HASSAN, A.; SABHA, M. Feature extraction for image analysis and detection using machine learning techniques. **International Journal of Advanced Networking and Applications**, v. 14, p. 5499–5508, 01 2023. Citation on page 38.

HE, K.; ZHANG, X.; REN, S.; SUN, J. **Deep Residual Learning for Image Recognition**. 2015. Available: <https://arxiv.org/abs/1512.03385>. Citation on page 68.

HEIMBERGER, M.; HORGAN, J.; HUGHES, C.; MCDONALD, J.; YOGAMANI, S. Computer vision in automated parking systems: Design, implementation and challenges. **arXiv preprint arXiv:2104.12537**, 2021. Citation on page 26.

IBM. **Object Detection**. 2025. Acesso em: 26 mar. 2025. Available: <https://www.ibm.com/think/topics/object-detection>. Citation on page 38.

____. **What is Cloud Computing?** 2025. Available: <https://www.ibm.com/think/topics/cloud-computing/>. Accessed: 28/03/2025. Citation on page 54.

____. **What is Computer Vision?** 2025. Available: <https://www.ibm.com/think/topics/computer-vision>. Accessed: 22/03/2025. Citation on page 30.

IBM Corporation. **Fine-tuning Pre-trained Models in Machine Learning**. 2025. <https://www.ibm.com/think/topics/fine-tuning>. Acessado em 5 de julho de 2025. Citation on page 43.

____. **O que é quantização?** 2025. <https://www.ibm.com/br-pt/think/topics/quantization>. Acessado em 7 de julho de 2025. Citation on page 38.

____. **Overfitting in Machine Learning: What Is It and How to Prevent It**. 2025. <https://www.ibm.com/br-pt/think/topics/overfitting>. Acessado em 5 de julho de 2025. Citation on page 43.

JIAO, L.; ZHANG, F.; LIU, F.; YANG, S.; LI, L.; FENG, Z.; QU, R. A survey of deep learning-based object detection. **IEEE Access**, Institute of Electrical and Electronics Engineers (IEEE), v. 7, p. 128837–128868, 2019. ISSN 2169-3536. Available: <http://dx.doi.org/10.1109/ACCESS.2019.2939201>. Citation on page 39.

JOCHER, G. *et al.* **YOLOv5**. 2020. <https://github.com/ultralytics/yolov5>. GitHub repository. Citation on page 45.

KABIR, M. M.; RAHMAN, A.; HASAN, M. N.; MRIDHA, M. Computer vision algorithms in healthcare: Recent advancements and future challenges. **Computers in Biology and Medicine**, v. 185, p. 109531, 2025. ISSN 0010-4825. Available: <https://www.sciencedirect.com/science/article/pii/S0010482524016160>. Citations on pages 36 and 37.

KALE, K.; PAWAR, S.; DHULEKAR, P. Moving object tracking using optical flow and motion vector estimation. In: **2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions)**. [S.l.: s.n.], 2015. p. 1–6. Citation on page 44.

KARYPIDIS, E.; MOUSLECH, S. G.; SKOULARIKI, K.; GAZIS, A. Comparison analysis of traditional machine learning and deep learning techniques for data and image classification. **WSEAS TRANSACTIONS ON MATHEMATICS**, World Scientific and Engineering Academy and Society (WSEAS), v. 21, p. 122–130, Mar. 2022. ISSN 1109-2769. Available: <http://dx.doi.org/10.37394/23206.2022.21.19>. Citation on page 43.

LIN, T.-Y.; DOLLáR, P.; GIRSHICK, R.; HE, K.; HARIHARAN, B.; BELONGIE, S. **Feature Pyramid Networks for Object Detection**. 2017. Available: <https://arxiv.org/abs/1612.03144>. Citation on page 68.

LIU, S.; QI, L.; QIN, H.; SHI, J.; JIA, J. **Path Aggregation Network for Instance Segmentation**. 2018. Available: <https://arxiv.org/abs/1803.01534>. Citation on page 46.

LIU, W.; ANGUELOV, D.; ERHAN, D.; SZEGEDY, C.; REED, S.; FU, C.-Y.; BERG, A. C. Ssd: Single shot multibox detector. In: ____. **Computer Vision – ECCV 2016**. Springer International Publishing, 2016. p. 21–37. ISBN 9783319464480. Available: <http://dx.doi.org/10.1007/978-3-319-46448-0_2>. Citation on page 40.

LIU, W.; WANG, S.; DENG, Z.; FAN, T.; JIA, M.; CHEN, H. A review of image feature descriptors in visual positioning. In: PéREZ-NAVARRO, A.; MONTOLIU, R.; TORRES-SOSPEDRA, J. (Ed.). **WiP Proceedings of the Eleventh International Conference on Indoor Positioning and Indoor Navigation - Work-in-Progress Papers (IPIN-WiP 2021) co-located with 11th International Conference on Indoor Positioning and Indoor Navigation (IPIN 2021), Lloret de Mar, Spain, 29 November - 2 December, 2021**. CEUR-WS.org, 2021. (CEUR Workshop Proceedings, v. 3097). Available: <http://ceur-ws.org/Vol-3097/paper19.pdf>. Citation on page 44.

MathWorks, Inc. **Object Detection – Computer Vision Toolbox (Deep Learning and Machine Learning Approaches)**. 2025. <https://www.mathworks.com/help/vision/object-detection.html>. Acessado em 5 de julho de 2025. Citation on page 39.

MELL, P.; GRANCE, T. **The NIST Definition of Cloud Computing**. [S.l.]: Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD, 2011. Citation on page 56.

MERZ, G.; LIU, Y.; BURKE, C. J.; ALEO, P. D.; LIU, X.; KIND, M. C.; KINDRATENKO, V.; LIU, Y. Detection, instance segmentation, and classification for astronomical surveys with deep learning deepdisc: detectron2 implementation and demonstration with hyper suprime-cam data. **Monthly Notices of the Royal Astronomical Society**, Oxford University Press (OUP), v. 526, n. 1, p. 1122–1137, Sep. 2023. ISSN 1365-2966. Available: <http://dx.doi.org/10.1093/mnras/stad2785>. Citation on page 40.

MICROSOFT. **What is Cloud Computing?** 2025. Available: <https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-cloud-computing>. Accessed: 27/03/2025. Citation on page 52.

____. **What is Computer Vision?** 2025. Available: <https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-computer-vision>. Accessed: 22/03/2025. Citation on page 30.

Microsoft Azure. **Azure Computer Vision Pricing**. 2025. Accessed: 2025-04-02. Available: <https://azure.microsoft.com/en-us/pricing/details/cognitive-services/computer-vision/>. Citations on pages 66 and 77.

NAUMANN, A.; HERTLEIN, F.; DöRR, L.; THOMA, S.; FURMANS, K. Literature review: Computer vision applications in transportation logistics and warehousing. **arXiv preprint arXiv:2304.06009**, 2023. Citation on page 26.

NVIDIA. **Performance Comparison**. 2025. Available: <https://docs.nvidia.com/vpi/bench_comparison.html>. Accessed: 24/03/2025. Citation on page 31.

NVIDIA Corporation. **Sistemas Embarcados | NVIDIA**. 2025. <https://www.nvidia.com/pt-br/autonomous-machines/embedded-systems/>. Acessado em 7 de julho de 2025. Citation on page 25.

OLIVEIRA, F.; COSTA, D. G.; ASSIS, F.; SILVA, I. Internet of intelligent things: A convergence of embedded systems, edge computing and machine learning. **Internet of Things**, v. 26, p. 101153, 2024. ISSN 2542-6605. Available: <https://www.sciencedirect.com/science/article/pii/S2542660524000945>. Citation on page 74.

OSHANA, R. 2 - overview of embedded systems and real-time systems. In: OSHANA, R. (Ed.). **DSP Software Development Techniques for Embedded and Real-Time Systems**. Burlington: Newnes, 2006, (Embedded Technology). p. 19–34. ISBN 978-0-7506-7759-2. Available: <https://www.sciencedirect.com/science/article/pii/B9780750677592500041>. Citation on page 73.

OSUNA, E.; FREUND, R.; GIROSIT, F. Training support vector machines: an application to face detection. In: **Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 1997. p. 130–136. Citation on page 42.

PANDA, A.; PANIGRAHI, D.; MITRA, S.; MITTAL, S.; RAHIMI, S. **Transfer Learning Applied to Computer Vision Problems: Survey on Current Progress, Limitations, and Opportunities**. 2024. Available: <https://arxiv.org/abs/2409.07736>. Citations on pages 41 and 42.

PARAB, C. U.; MWITTA, C.; HAYES, M.; SCHMIDT, J. M.; RILEY, D.; FUE, K.; BHANDARKAR, S.; RAINS, G. C. Comparison of single-shot and two-shot deep neural network models for whitefly detection in iot web application. **AgriEngineering**, v. 4, n. 2, p. 507–522,

2022. ISSN 2624-7402. Available: <https://www.mdpi.com/2624-7402/4/2/34>. Citations on pages 40 and 68.

PARVEZ, I.; RAHMATI, A.; GUVENC, I.; SARWAT, A. I.; DAI, H. **A Survey on Low Latency Towards 5G: RAN, Core Network and Caching Solutions**. 2018. Available: <https://arxiv.org/abs/1708.02562>. Citation on page 76.

PETCHIAPPAN, T. The hidden costs of cloud security based on understanding financial implications for businesses. **Journal of Contemporary Education, Theory and Artificial Intelligence**, 2024. JCETAI-114. Received: 03 Oct 2024; Accepted: 11 Oct 2024; Published: 18 Oct 2024. Citations on pages 112 and 114.

PLATZER, D. Regarding the pain of users: Towards a genealogy of the "pain point". In: **Ethnographic Praxis in Industry Conference Proceedings**. [S.l.]: American Anthropological Association, 2018. v. 2018, n. 1, p. 301–315. Citation on page 25.

POTTER, M. C.; WYBLE, B.; HAGMANN, C. E.; MCCOURT, E. S. Detecting meaning in rsvp at 13 ms per picture. **Attention, perception and psychophysics**, v. 76, n. 2, p. 270–279, 2014. Citation on page 31.

POURSAEED, O.; MATERA, T.; BELONGIE, S. Vision-based real estate price estimation. **arXiv preprint**, 2017. Citation on page 26.

Python Software Foundation. **Built-in Types — Sequence Types**. 2025. <https://docs.python.org/3/library/stdtypes.html>. Acessado em 7 de julho de 2025. Citation on page 38.

REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. You only look once: Unified, real-time object detection. **arXiv preprint arXiv:1506.02640**, 2016. Citations on pages 44 and 45.

REN, S.; HE, K.; GIRSHICK, R.; SUN, J. **Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks**. 2016. Available: <https://arxiv.org/abs/1506.01497>. Citation on page 39.

RIBEIRO, V. N.; HIRATA, N. S. T. **Efficient License Plate Recognition in Videos Using Visual Rhythm and Accumulative Line Analysis**. 2025. Available: <https://arxiv.org/abs/2501.04750>. Citation on page 34.

ROH, Y.; HEO, G.; WHANG, S. E. A survey on data collection for machine learning: A big data - ai integration perspective. **IEEE Transactions on Knowledge and Data Engineering**, v. 33, n. 4, p. 1328–1347, 2021. Citation on page 32.

SATYANARAYANAN, M. The emergence of edge computing. **Computer**, v. 50, n. 1, p. 30–39, 2017. Citation on page 74.

SHI, W.; CAO, J.; ZHANG, Q.; LI, Y.; XU, L. Edge computing: Vision and challenges. **IEEE Internet of Things Journal**, v. 3, n. 5, p. 637–646, 2016. Citation on page 73.

SHORTEN, C.; KHOSHGOFTAAR, T. M. A survey on image data augmentation for deep learning. **Journal of Big Data**, v. 6, n. 1, p. 60, 2019. ISSN 2196-1115. Available: <https://doi.org/10.1186/s40537-019-0197-0>. Citation on page 46.

SINDAGI, V. A.; PATEL, V. M. A survey of recent advances in cnn-based single image crowd counting and density estimation. **Pattern Recognition Letters**, v. 107, p. 3–16, 2018. Citation on page 26.

SINGH, J.; URVASHI; SINGH, G.; MAHESHWARI, S. Augmented reality technology: Current applications, challenges and its future. In: **2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA)**. [S.l.: s.n.], 2022. p. 1722–1726. Citation on page 35.

Startups Brasil. **Proptech SuaQuadra levanta seed de R$ 21 milhões com a Kaszek**. 2024. Acesso em: 6 jun. 2025. Available: <https://startups.com.br/negocios/rodada-de-investimento/proptech-suaquadra-levanta-seed-de-r-21-milhoes-com-a-kaszek/>. Citation on page 26.

Statista. **Worldwide Market Share of Leading Cloud Infrastructure Service Providers**. 2025. Available: <https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/>. Citation on page 51.

Synergy Research Group. **Cloud Market Jumped to $330 Billion in 2024 – GenAI is Now Driving Half of the Growth**. 2025. Available: <https://www.srgresearch.com/articles/cloud-market-jumped-to-330-billion-in-2024-genai-is-now-driving-half-of-the-growth>. Citation on page 52.

TURING, A. M. I.—computing machinery and intelligence. **Mind**, LIX, n. 236, p. 433–460, 10 1950. ISSN 0026-4423. Available: <https://doi.org/10.1093/mind/LIX.236.433>. Citation on page 31.

ULTRALYTICS. **YOLOv8: Next-Generation Object Detection and Segmentation**. 2023. <https://github.com/ultralytics/ultralytics>. Citation on page 45.

____. **Ultralytics YOLO Models Documentation**. 2024. Acesso em: 7 abr. 2025. Available: <https://docs.ultralytics.com/models/>. Citation on page 47.

____. **Ultralytics YOLO on Raspberry Pi**. 2024. Acesso em: 7 abr. 2025. Available: <https://docs.ultralytics.com/guides/raspberry-pi/>. Citation on page 81.

____. **YOLOv11: New state-of-the-art real-time object detection model**. 2024. <https://github.com/ultralytics/ultralytics>. Citation on page 45.

Ultralytics. **Performance Metrics for Object Detection Models: mAP, Speed, Params, FLOPs**. 2025. <https://docs.ultralytics.com/guides/yolo-performance-metrics/>. Acessado em 5 de julho de 2025. Citation on page 47.

____. **Ultralytics YOLO Architecture: CSPNet Backbone, PANet Neck, Data Augmentation, Anchors and Detection Head**. 2025. <https://docs.ultralytics.com/pt/yolov5/tutorials/architecture_description/>. Acessado em 5 de julho de 2025. Citation on page 45.

WANG, C.-Y.; LIAO, H.-Y. M.; YEH, I.-H.; WU, Y.-H.; CHEN, P.-Y.; HSIEH, J.-W. **CSPNet: A New Backbone that can Enhance Learning Capability of CNN**. 2019. Available: <https://arxiv.org/abs/1911.11929>. Citation on page 45.

WOJKE, N.; BEWLEY, A.; PAULUS, D. Simple online and realtime tracking with a deep association metric. In: **2017 IEEE International Conference on Image Processing (ICIP)**. [s.n.], 2017. p. 3645–3649. Available: <https://ieeexplore.ieee.org/document/8296962>. Citations on pages 48, 49, and 50.

ZHANG, J.; AN, D.; ZHANG, Y.; WANG, X.; WANG, X.; WANG, Q.; PAN, Z.; YUE, Y. A review on face mask recognition. **Sensors**, v. 25, n. 2, 2025. ISSN 1424-8220. Available: <https://www.mdpi.com/1424-8220/25/2/387>. Citation on page 41.

ZHANG, Q.; CHENG, L.; BOUTABA, R. Cloud computing: State-of-the-art and research challenges. **Journal of Internet Services and Applications**, v. 1, p. 7–18, 05 2010. Citation on page 56.


ICMC
USP
SÃO CARLOS