eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Department of Computer Sciences and Artificial Intelligence

Optimization for Deep Learning Systems Applied to Computer Vision

by:

David Montero Martín

1. Supervisor Ph.D. Naiara Aginako Bengoa
Department of Computer Sciences and Artificial Intelligence
University of the Basque Country (UPV/EHU)

2. Supervisor Ph.D. Marcos Nieto Doncel
Connected & Cooperative Automated Systems
Vicomtech

David Montero Martín

Optimization for Deep Learning Systems Applied to Computer Vision

Supervisors: Ph.D. Naiara Aginako Bengoa and Ph.D. Marcos Nieto Doncel

University of the Basque Country

Department of Computer Sciences and Artificial Intelligence

Vicomtech

Connected & Cooperative Automated Systems

Donostia - San Sebastián

Abstract

Computer Vision (CV) is present everywhere around us; we live with this technology on a daily basis. It can be found in our mobile phones, computers, cars, alarm systems, etc. It has contributed to many advances in our society. For instance, CV is one of the foundational technologies of industrial automation. It has helped improve product quality, speed production, and streamline manufacturing and logistics for decades. CV has also been an essential part of advances in autonomous driving, providing algorithms that help solve complex problems such as path planning, driving scene perception, and behavior arbitration.

Since the Deep Learning (DL) revolution and especially over the last years (2010-2022), Deep Neural Networks (DNNs) have become an essential part of the CV field, and they are present in all its sub-fields (video-surveillance, industrial manufacturing, autonomous driving, ...) and in almost every new state-of-the-art application. However, DNNs are very complex and the architecture needs to be carefully selected and adapted in order to maximize its efficiency and effectiveness. On many occasions, once the desired architecture has been selected, it is slightly tuned and integrated into the system to start its operation. However, often this is not enough to achieve maximum performance. It is not sufficient to just select the appropriate architecture: it is necessary to adapt it to the specific use case and to the rest of the system components since it has not been specifically developed for it. And not only the network must be adapted to the system, but also the rest of the components that interact with the network, such as the preprocessing or postprocessing algorithms. Such proper optimizations, systems based on DNNs can achieve an important boost in performance. And this performance improvement does not refer only to an increase in precision, but can also mean a reduction in the memory necessary to execute the process and/or its processing speed.

This thesis aims at providing knowledge and tools for the optimization of systems based on Deep Learning applied to different real use cases within the field of Computer Vision, in order to maximize their effectiveness and efficiency. The thesis is supported by six main publications that contribute to a series of objectives grouped into two research lines: the optimization of the DNNs and the optimization of the CV systems.

Acknowledgement

Thanks to all the people who have helped me grow both personally and professionally over the years.

Thanks to everyone who has encouraged me to do my thesis and to those who have supported me in difficult times.

Thanks to all my co-authors for contributing with their talent and effort to the writing and publication of the articles that support this thesis.

Without all of you, this would not have been possible.

Thank you from the bottom of my heart.

Contents

1 Introduction

Computer Vision (CV) is a field of Artificial Intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs, and take actions or make recommendations based on that information. If AI enables computers to think, CV enables them to see, observe and understand.

CV is present everywhere around us, we live with this technology on a daily basis. It can be found in our mobile phones, computers, cars, alarm systems, etc. It has also contributed to many advances in our society. For instance, CV is one of the foundational technologies of industrial automation. It has helped improve product quality, speed production, and streamline manufacturing and logistics for decades. CV has also been an essential part of advances in autonomous driving, providing algorithms that help solve complex problems such as path planning, driving scene perception, and behavior arbitration.

All these advances have been driven by the appearance and application of different technologies. One of the most important, and which has been gaining ground in recent years, is Deep Learning (DL). DL is a subclass of Machine Learning (ML) in AI based on artificial neural networks (ANNs) with representation learning. Learning can be supervised, semi-supervised or unsupervised. ANNs were inspired by information processing and distributed communication nodes in biological systems. However, ANNs have various differences from biological brains. Specifically, artificial neural networks tend to be static and symbolic, while the biological brain of most living organisms is dynamic (plastic) and analogue [MWK16; Ben+15]. Then, when we stack multiple layers between the input and output layers, the ANN is called deep neural network (DNN) [Ben07; Sch15].

The first general working learning algorithm for supervised, deep, feedforward, multilayer neural networks was published by Alexey Ivakhnenko and Lapa in 1967 [ILM67]. Nevertheless, the term Deep Learning was not introduced to the ML community until by Rina Dechter in 1986 [Dec86], and the ANN until 2000 by Igor Aizenberg and colleagues, in the context of Boolean threshold neurons[AAV12; GS05]. Since then, significant advances were achieved, such as the introduction of the Convolutional Neural Networks (CNNs)[Zha+88; Zha+90], or the recurrent layers (for instance the Long Short-Term Memory (LSTM) layer[HS97]).

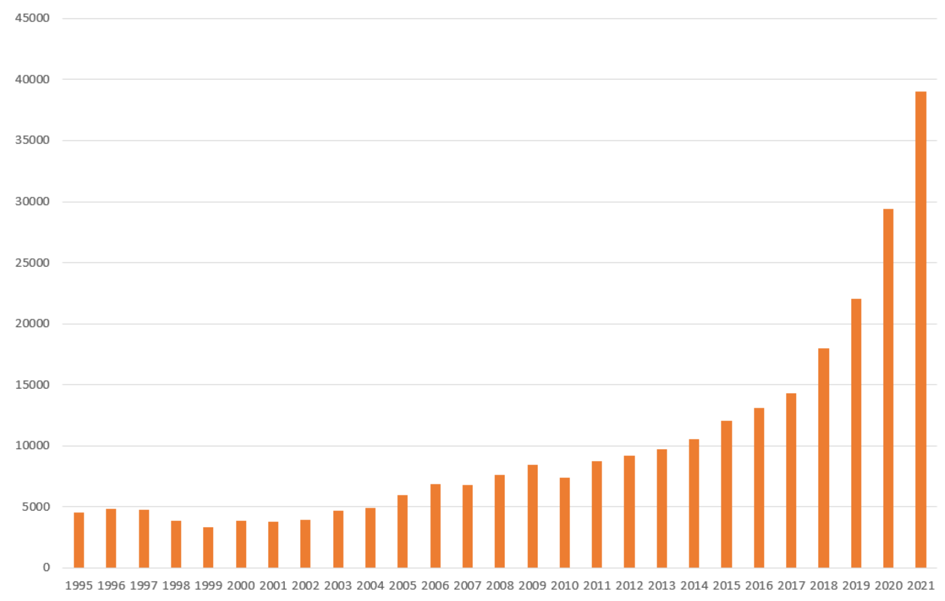


Fig. 1.1: Publications per year related to Neural Networks in Sciencedirect

The impact of DL in the industry started in the early 2000s, when it was estimated that the CNNs already processed between 10% and 20% of all the checks written in the United States. However, due to the high computational cost and the lack of specialized hardware, DL did not become truly popular within the field of CV until the decade of 2010. In 2009, Nvidia was involved in what was called the "big bang" of DL, as deep neural networks were trained and deployed using Nvidia graphics processing units (GPUs). That year, Andrew Ng, a renowned ML researcher, announced that GPUs could increase the speed of DL systems by about 100 times. In particular, GPUs are well-suited for the matrix/vector computations involved in machine learning[OJ04; MV19; CPS06]. GPUs speed up training algorithms by orders of magnitude, reducing running times from weeks to days[Cir+10; RMN09]. In addition, specialized hardware and algorithm optimizations could be used for efficient processing of DL models[Sze+17]. These important advances led to new fast implementations of Convolutional Neural Networks (CNNs) on GPUs [CMS12; KSH12; Cir+13] and increased the interest of the scientific community in DNNs, which was reflected in the number of publications (see Figure 1.1).

Since the DL revolution and especially over the last years (2010-2022), DNNs have become an essential part of the CV field, and they are present in all its sub-fields (video-surveillance, industrial manufacturing, autonomous driving, ...) and in almost every new state-of-the-art application that is developed. However, DNNs are very complex and the architecture needs to be carefully selected and adapted in order to maximize its efficiency. In many cases, networks are not specifically designed for the considered use case, they are simply recycled from other applications and slightly adapted, without taking into account the particularities of the use case or the interaction with the rest of the system components, which usually results in a performance drop.

This research work aims at providing knowledge and tools for the optimization of systems based on Deep Learning applied to different real use cases within the field of Computer Vision, in order to maximize their effectiveness and efficiency.

1.1 Context of this research work

This Ph.D. dissertation is supported by a compendium of articles resulting from the cooperation between researchers from Vicomtech research center and the University of the Basque Country during the last four years.

Vicomtech is a technological research center specialized in AI, CV, and Interaction. Although it also leads and collaborates in pure research projects, Vicomtech's main objective is to build a bridge between basic research and industry, developing real solutions for companies. For this reason, many of the innovations presented in this thesis are highly focused on concrete and practical applications. However, much of this knowledge can also be extrapolated to other applications within the field of CV and even, at a more general level, within the field of DL.

Regarding its internal structure, Vicomtech is divided into several departments according to its specialization into the different branches of the industry. Among all of them, the contributions of this thesis have been obtained mainly by collaborating on projects within three departments:

- Connected and Cooperative Automated Systems: focuses on projects related to the automotive industry, such as autonomous driving, collision avoidance, or pedestrian detection.

- Intelligent Security Video Analytics: conducts projects related to video surveillance and scene analysis and monitoring, using advanced techniques such as human detection and re-identification or face recognition.

- Intelligent Systems for Mobility and Logistics: tackles projects that require managing large surfaces and/or important amounts of data in an efficient and scalable manner.

On the other hand, collaborations with the University of the Basque Country have taken place within the department of Computer Science and Artificial Intelligence.

Although they are within the field of CV, the three departments work on projects that are very varied. However, most of these projects have in common the need to use systems based on DL to achieve competitive solutions that offer state-of-the-art results. For example, many automotive-related projects require DL models to detect vehicles, pedestrians, or other obstacles on the road. Or in the case of video surveillance, the most advanced models for the detection and re-identification of people and faces are also based on this technology. This need is the link between all these projects and has led to all the articles that make up this thesis.

1.2 Motivation

Since the DL revolution and especially over the last years, DNNs have become an essential part within the CV field, and they are present in all its sub-fields (video-surveillance, industrial manufacturing, autonomous driving, ...) and in almost every new state-of-the-art application. For example, in object detection, one of the best-known tasks in artificial vision, models based on DNNs have highly outperformed traditional models [Zha+19b; WSH20; Xia+20], and day by day they continue to improve and to achieve better results.

However, in most cases, these detection models are evaluated on reference datasets, with specific classes and good quality images and annotations [Lin+14; Den+09; Eve+10]. These datasets try to cover as many different types of objects as possible and to generalize to any type of data. However, usually the results are worse when applied to other use cases, with a different quality of the data and objects with different characteristics (range of sizes, shape, color, ...). The same problem can be extrapolated for other CV tasks, such as object classification or image segmentation.

Consequently, it may happen that an architecture that performs worse than another in one or more reference datasets, is better adapted to the data of a particular project and obtains better results. Therefore, to choose the right architecture, many factors must be taken into account. First, an in-depth analysis of the use case and the available data must be carried out to extract a list of necessary requirements or qualities that the network must meet (wide range of object sizes, noise resistance, precision...). With this list, a more specific search among the available architectures can be performed in order to select the one that best suits the specific use case. For example, if the objects have a small range of sizes, the number of architectures that could adapt well to the data will be greater and the search could focus more on obtaining the best precision [WBL22; TPL20; Liu+21]. However, if the range of sizes is very wide, it would be necessary to search architectures that were specifically designed for that problem, such as those based on Feature Pyramid Networks [Lin+17a] or Cascade Networks [XZ17].

Apart from precision, when selecting the architecture, the processing speed of the network must also be taken into account. Usually, the improvement in the precision of the new models is accompanied by an increase in the number of parameters and in the computation time, as it happens with the different architectures within each YOLO version [BWL20b; Joc+21; Li+22] or with the different Efficientnet backbones [TL19]. Processing speed is very important in many applications, especially in those that are focused on operating in real-time. In these cases, the best trade-off between precision and speed is always sought, although this usually means not reaching one of the two initially proposed requirements (or neither of them).

The DNN training is another very important part of the process and is highly dependent on the type of data and the use case [Zha+19a]. The choice of suitable preprocessing steps can have a great impact on the results (cropping, resizing, normalization, batching,...). For example, in facial recognition, the use of face alignment based on facial landmarks makes the model converge faster and brings an improvement in accuracy [KKG07]. Furthermore, depending on the quality and the diversity of the data, and on the task difficulty, data augmentation techniques can boost the results and avoid overfitting [SK19]. These techniques range from the simplest based on random cropping or rotation on the existing data to the most advanced, such as the generation of synthetic data [EPS17]. Another important design choice for the training process is the loss function. It highly depends on the type of task, but also on the type of data. For instance, in the object detection task, the Weighted Cross-Entropy loss [PY20] can help fight class imbalance, and the Focal Loss [Lin+17b] can improve the results for dense sampling strategies.

On many occasions, once the desired architecture has been selected and trained, it is slightly tuned and integrated into the system with the rest of the components to start its operation. However, in most cases, this is not enough to achieve maximum network performance. It is not sufficient just to select the appropriate architecture, it is necessary to adapt it to the specific use case and to the rest of the system components since it has not been specifically developed for it. And not only the network must be adapted to the system, but also the rest of the components that interact with the network, such as the preprocessing or postprocessing algorithms. With proper optimizations, systems based on DNNs can achieve an important boost in performance. And this performance improvement does not refer only to an increase in precision, but can also mean a reduction in the memory necessary to execute the process and/or its processing speed.

For instance, in [SWC20], the authors apply a series of optimization to a well-known keypoints-detection network, OpenPose [Cao+21], in order to improve its performance on dedicated hardware. With these optimizations, they report a speedup of 11.5x in the end-to-end performance. Another example of the impact of the network optimization is given in [Xie+20], where the authors perform channel pruning to reduce the number of parameters of the convolution layers of an object detection network. More specifically, they propose a localization-aware auxiliary network to find out the channels with key information for classification and regression so that they can conduct channel pruning directly for object detection, which saves lots of time and computing resources. They evaluate their approach on MS COCO dataset, where they are able to prune 70% parameters of the model with just a small drop in accuracy.

On the other hand, there can also be found examples about the benefits of optimizing other system components. As an example, in [Ces+20], the authors evaluate several common Python libraries which are used for image preprocessing and analyze the impact of different approaches with respect to central processing unit (CPU) usage. They report big differences between the different approaches in the processing time, reflecting the importance of the preprocessing optimization. In [Son+19], the authors focus their efforts on optimizing the Non-Maximum Suppression (NMS) postprocessing algorithm. They combine a harmony search algorithm with NMS to improve its capability to perceive nearby objects in cluttered scenes. They evaluate the performance of their algorithm with two different detection networks, showing an improvement in the average precision of these two detection networks. Moreover, the location performance and average recall of these two detectors are also improved.

All these examples show the importance of optimizing neural networks and the different components within systems based on Deep Learning, and their impact on all areas of system performance. Often, without optimizations, the solution cannot be effectively used, and thus we can consider optimization as a mandatory step in modern DL-based CV systems for real-life applications. This motivates the objective of this thesis: to provide knowledge and tools for the optimization of systems based on Deep Learning applied to different real use cases within the field of Computer Vision, in order to maximize their effectiveness and efficiency.

1.3 Hypothesis

Considering the discussion and the references provided in the previous section, the following hypotheses are formulated, and they will serve as the basis for the research carried out in this thesis:

1. The selection of the neural network architecture is a crucial step in the creation of a DL-based CV system.

   a) This choice cannot be based uniquely on a comparison of the results obtained in public benchmarks, since the performance of the architecture can greatly vary depending on the quality and type of data.

   b) Different factors must be taken into account, and the importance of each one varies depending on the use case (accuracy, memory consumption, processing time,...)

2. The definition of the training process for the DNN has a great impact on the quality of the results. It must be designed very carefully attending to the type of data and the use case.

   a) The appropriate preprocessing techniques must be chosen for the type of data (cropping, resizing, normalization, batching,...) [KKG07].

   b) Depending on the quantity and quality of the data, data augmentation techniques (random cropping, brightness, rotation,...) should be considered [SK19], and even the generation of additional synthetic data [EPS17].

   c) Selecting the appropriate loss function is also very important and dependent on the data (weighted Cross-Entropy [PY20], Focal Loss [Lin+17b], ...).

3. Optimizing the neural network architecture for a particular use case can greatly increase system performance [SWC20; Xie+20]. There are different ways to optimize a neural network:

   a) Reduction of the number of parameters through techniques such as weight pruning [Rum+20] or knowledge distillation [Gou+21]. These improvements impact memory consumption, processing time, and even accuracy.

   b) Hardware-based optimizations, such as the framework selection (TensorRT [Jeo+22], OpenVINO [Dem+21],...) or the weights quantization [LBL19]. These improvements also impact memory consumption and processing time.

   c) Other optimizations focused on improving the accuracy, like the use of multi-tasking to enhance the performance on the main task [Lu+20].

4. At the same time, adapting and optimizing the rest of the system components to work in harmony with the neural network can bring important improvements to the whole system [Ces+20; Son+19; Nie+14].

    a) Preprocessing and postprocessing algorithms are often more time-consuming than the DNN itself and can be highly optimized.

    b) Improving the postprocessing algorithm and adapting it to the use case can have a high impact on the accuracy.

1.4 Objectives

Attending to the hypotheses formulated above, two research lines are defined, with several objectives associated with each one.

1.4.1 Optimization of Deep Neural Networks

This research line is focused on maximizing the performance of the DNNs applied to specific use cases. This optimization process includes the selection of the most suitable architecture, the design of the training pipeline, and the modifications and optimizations applied before and after it. More specifically, the objectives pursued within this research line are the following:

Obj 1.1 Study the latest DNN architectures and find the optimal ones for different use cases. To accomplish this task, public and private benchmarks will be used and different performance factors will be measured, where the importance of each one will depend on the considered use case.

Obj 1.2 Design training pipelines customized for the different projects. These pipelines must adapt to the available data and try to overcome its limitations (shortage of data, homogeneous data, noisy annotations, class imbalance,...). Above all, the study will focus on the following parts of the process:

- Selecting the appropriate dataset (or datasets).

- Conducting an exploratory data analysis (EDA) in order to understand the data and its weaknesses, and to help define the rest of the stages of the pipeline.

- Data curation: cleaning and preparing the data (for instance, to remove duplicated data, wrong annotations, ...).

- Applying data augmentation techniques if necessary.

- Preparing the most suitable preprocessing techniques for the data.

- Selecting the optimal optimizer, loss function, and hyperparameters for the training process.

- Choosing the right metrics to monitor and validate the experiments, as well as selecting the best weights.

Obj 1.3 Optimize DNN architectures, both before and after training, to increase their speed and accuracy and reduce the number of parameters and resource consumption. To address this objective, state-of-the-art techniques present in the literature will be used and new ones will be proposed. These techniques can be divided into two groups:

- Optimization techniques prior to the training process. This group includes procedures such as reducing the number or size of the layers, or the use of multi-task architectures to join two main tasks or add a secondary one that supports the main one.

- Post-training optimization techniques. Within this category, well-known techniques can be found, such as weight pruning or weight quantization, as well as the procedure of porting the architecture to frameworks dedicated to inference, such as TensorRT or OpenVINO.

1.4.2 Optimization of Computer Vision Systems

This research line seeks to maximize the performance of the CV system, attending to the design at a global level, but also to the rest of the components that interact with the neural network. More specifically, the optimization will be focused on the following system components:

Obj 2.1 Data management: searching for an efficient handling of the images and the rest of the input data.

Obj 2.2 Data preprocessing: selecting the optimal data preprocessing functions (resize, normalization, cropping, etc.), taking into account the previous training step of the neural network, as well as parallelizing and combining them to reduce latency and resource consumption as much as possible.

Obj 2.3 Transition components between neural networks (in systems that require the use of more than one network): designed to seek optimal use of resources and reduce bottlenecks in system processing time.

Obj 2.4 Postprocessing algorithms: selecting, adapting, and optimizing the appropriate postprocessing algorithms depending on the use case (tracking, clustering, etc).

1.5 Main publications

This thesis is supported by 6 main publications that contribute to the different research lines and objectives defined in the previous section (see Table 1.1). In 5 of them, the author of this thesis is presented as the main author and in the remaining one as the second author. Furthermore, 3 publications have been published in journals in the first or second quartile and the other 3 have been presented in international conferences.

Tab. 1.1: List of the main publications that support the thesis along with the type of publication, the author position, and the tackled objectives.

Publication	Type	Author pos	Objectives
[Mon+22a]	Journal Q1	First	Obj2.4
[YMA21]	Journal Q1	Second	Obj1.1, Obj1.2, Obj1.3
[Mon+on]	Journal Q2	First	Obj1.2, Obj1.3, Obj2.4
[Mon+21]	Conf GGS rating B	First	Obj1.3, Obj2.1, Obj2.3, Obj2.4
[Mon+22b]	Conference	First	Obj1.2, Obj1.3
[Mon+19]	Conference	First	Obj2.4

[Mon+22a] Montero, D., Aginako, N., Sierra, B., & Nieto, M. (2022). *Efficient Large-Scale Face Clustering Using an Online Mixture of Gaussians*. Eng. Appl. Artif. Intell., 114, 105079. (see Section 4.1)

Abstract: In recent years, the number of applications demanding real-time face clustering algorithms has increased, especially for security and surveillance purposes. However, state-of-the-art face clustering methods are offline, they need to repeat the whole clustering process every time new data arrives, and thus, they are not

suitable for real-time applications. On the other hand, online clustering methods are highly dependent on the order and the size of the data, and they are less accurate than offline methods. To overcome these limitations, we present an online gaussian mixture-based clustering method (OGMC). The key idea of this method is the proposal that an identity can be represented by more than just one distribution or cluster. Using feature vectors extracted from the incoming faces, OGMC generates clusters that may be connected to others depending on their proximity and their robustness, and updates their connections every time their parameters are updated. With this approach, we reduce the dependency of the clustering process on the order and the size of the data and we are able to deal with complex data distributions. Experimental results show that OGMC outperforms state-of-the-art clustering methods on large-scale face clustering benchmarks not only in accuracy, but also in efficiency and scalability.

[YMA21] Yebes, J.J., Montero, D., & Arriola, I. (2021). *Learning to Automatically Catch Potholes in Worldwide Road Scene Images*. IEEE Intelligent Transportation Systems Magazine, 13, 192-205. (see Section 4.2)

Abstract: Among several road hazards that are present in any paved way in the world, potholes are one of the most annoying and involving higher maintenance costs. There is an increasing interest on the automated detection of these hazards enabled by technological and research progress. Our work tackled the challenge of pothole detection from images of real world road scenes. The main novelty resides on the application of latest progress in Artificial Intelligence to learn the visual appearance of potholes. We built a large dataset of images with pothole annotations. They contained road scenes from different cities in the world, taken with different cameras, vehicles and viewpoints under varied environmental conditions. Then, we fine-tuned four different object detection models based on Deep Neural Networks. We achieved mean average precision above 75% and we used the pothole detector on the Nvidia DrivePX2 platform running at 5-6 frames per second. Moreover, it was deployed on a real vehicle driving at speeds below 60 km/h to notify the detected potholes to a given Internet of Things platform as part of AUTOPILOT H2020 project.

[Mon+on] Montero, D., Aranjuelo, N., Leskovsky, P., Loyo, E., Nieto, M. & Aginako, N. (2023). *Multi-Camera BEV Video-Surveillance System for Efficient Monitoring of Social Distancing*. Multimedia Tools and Applications, pending publication. (see Section 4.3)

Abstract: The current sanitary emergency situation caused by COVID-19 has increased the interest in controlling the flow of people in indoor infrastructures, to ensure compliance with the established security measures. Top view camera-based solutions have proven to be an effective and non-invasive approach to accomplish this task. Nevertheless, current solutions suffer from scalability problems: they cover limited range areas to avoid dealing with occlusions and only work with single camera scenarios. To overcome these problems, we present an efficient and scalable people flow monitoring system that relies on three main pillars: an optimized top view human detection neural network based on YOLO-V4, capable of working with data from cameras at different heights; a multi-camera 3D detection projection and fusion procedure, which uses the camera calibration parameters for an accurate real-world positioning; and a tracking algorithm which jointly processes the 3D detections coming from all the cameras, allowing the traceability of individuals across the entire infrastructure. The conducted experiments show that the proposed system generates robust performance indicators and that it is suitable for real-time applications to control sanitary measures in large infrastructures. Furthermore, the proposed projection approach achieves an average positioning error below 0.2 meters, with an improvement of more than 4 times compared to other methods.

[Mon+21] Montero, D., Unzueta, L., Goenetxea, J., Aranjuelo, N., Loyo, E., Otaegui, O., & Nieto, M. (2021). *Multi-Stage Dynamic Batching and On-Demand I-Vector Clustering for Cost-effective Video Surveillance*. 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP). (see Section 4.4)

Abstract: In this paper, we present a cost-effective Video-Surveillance System (VSS) for face recognition and online clustering of unknown individuals at large scale. We aim to obtain Performance Indicators (PIs) for people flow monitoring in large infrastructures, without storing any biometric information. For this purpose, we focus on how to take advantage of a central GPU-enabled computing server, connected to a set of videosurveillance cameras, to automatically register new identities and update their descriptive data as they are reidentified. The proposed method comprises two main procedures executed in parallel. A Multi-Stage Dynamic Batching (MSDB) procedure efficiently extracts facial identity vectors (i-vectors) from captured images. At the same time, an On-Demand I-Vector Clustering (ODIVC) procedure clusters the i-vectors into identities. This clustering algorithm is designed to progressively adapt to the increasing data scale, with a lower decrease in its effectiveness compared to other alternatives. Experimental results show that ODIVC achieves state-ofthe-art results in well-known large scale datasets and that our VSS can detect, recognize and cluster in real time faces coming from up to 40 cameras with a central off-the-shelf GPU-enabled computing server.

[Mon+22b] Montero, D., Nieto, M., Leskovský, P., & Aginako, N. (2022). *Boosting Masked Face Recognition with Multi-Task ArcFace*. 16th International Conference on Signal Image Technology & Internet based Systems (SITIS). (see Section 4.5)

Abstract: In this article, we tackle the recognition of faces wearing surgical masks. Surgical masks have become a necessary piece of daily apparel because of the COVID-19-related worldwide health problem. Modern face recognition models are in trouble because they were not made to function with masked faces. Furthermore, in order to stop the infection from spreading, apps capable of detecting if the individuals are wearing masks are also required. To address these issues, we present an end-to-end approach for training face recognition models based on the ArcFace architecture, including various changes to the backbone and loss computation. We also use data augmentation to generate a masked version of the original dataset and mix them on the fly while training. Without incurring any additional computational costs, we modify the chosen network to output also the likelihood of wearing a mask. Thus, the face recognition loss and the mask-usage loss are merged to create a new function known as Multi-Task ArcFace (MTArcFace). The conducted experiments demonstrate that our method outperforms the baseline model results when faces with masks are considered, while achieving similar metrics on the original dataset. In addition, it obtains a 99.78% of mean accuracy in mask-usage classification.

[Mon+19] Montero, D., Aranjuelo, N., Senderos, O., & Nieto, M. (2019). *BEV Object Tracking for LIDAR-based Ground Truth Generation*. 2019 27th European Signal Processing Conference (EUSIPCO), 1-5. (see Section 4.6)

Abstract: Building ADAS (Advanced Driver Assistance Systems) or AD (Autonomous Driving) vehicles implies the acquisition of large volumes of data and a costly annotation process to create labeled metadata. Labels are then used for either ground truth composition (for test and validation of algorithms) or to set-up training datasets for machine learning processes. In this paper we present a 3D object tracking mechanism that operates on detections from point cloud sequences. It works in two steps: first an online phase which runs a Branch and Bound algorithm (BBA) to solve the association between detections and tracks, and a second filtering step which adds the required temporal smoothness. Results on KITTI dataset show the produced tracks are accurate and robust against noisy and missing detections, as produced by state-of-the-art Deep Learning detectors.

1.6 Thesis Structure

This thesis is divided into four chapters, which are described below:

- **Introduction**: provides some context about the conducted research work, and exposes the motivation, the hypothesis and the objectives of the thesis. In addition, the main publications are also listed.

- **Research results**: presents an analysis of the results and the contributions of each main publication to the different research lines and objectives defined in the introduction.

- **Conclusions**: in this chapter the conclusions obtained in the thesis are collected and new ideas are proposed that can be used to continue with the proposed research lines in the future.

- **Publications**: gathers all the main publications on which the thesis is based, as well as other secondary ones that are also relevant in its field.

Finally, the bibliography and the list of figures and tables are included at the end of the document.

Research results

2

In this chapter, the results obtained in the main research works that support the thesis are analyzed, as well as the contributions made related to the objectives of the different research lines defined in the previous chapter. For each one, an introduction is provided first about the motivation and the objectives in order to give some context. Then, a summary is presented about the development, the experiments, and the results, focusing on the parts that are more relevant to this thesis. Finally, the contributions of the work are highlighted and matched with their corresponding research lines. For full details about the methodology and the experiments, please refer to Section 4.

2.1 Efficient Large-Scale Face Clustering Using an Online Mixture of Gaussians

2.1.1 Motivation and objectives

In this work, we address the problem of large-scale online face clustering: given a continuous stream of unknown faces, create a database grouping the incoming faces by their identity. Furthermore, the application must meet the following requirements:

- The database must be updated every time a new face arrives.
- The solution must be efficient, accurate, and scalable.
- Data privacy must be granted: personal information that could be used to identify the subjects should not be stored [NSM05; Bow04; Erk+09].

To overcome the data privacy limitation, we rely on a face recognition model, a DNN that can infer feature vectors (f-vectors) from the targeted face images, which correspond to abstract representations of the people's appearances used for training [Den+19a; SKP15]. This way, the later clustering algorithm will work over these vectors and we do not need to store sensitive information about the subjects. An overview of the desired online clustering system is illustrated in Figure 2.1.

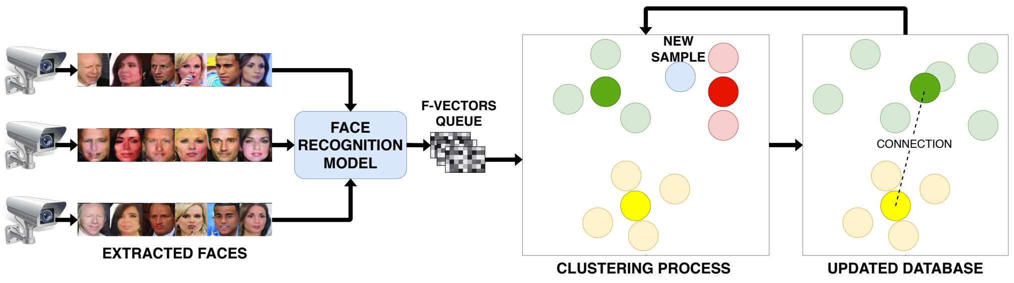


Fig. 2.1: Example of an online clustering system. A continuous stream of face images, extracted from a set of video-surveillance cameras, are processed by a face recognition model and the extracted f-vectors are enqueued. The online clustering process updates the database with every new sample without repeating the whole process.

Thus, we have two key components in the system, the face recognition model and the online clustering algorithm. In this work, we focus on the clustering algorithm, since we believe that it is the component that has the greatest room for improvement, given the peculiarities of the use case and the state-of-the-art in the literature. As an example of these peculiarities, these real-time applications may work in large-scale unconstrained environments, where no information about the distribution of face representations or the number of identities is available. Furthermore, the faces may come from multiple cameras placed in different locations and positions, so they may have different orientations, lighting conditions, partial occlusions, etc. This will lead to complex data distributions. Most traditional and state-of-the-art clustering methods are offline ([Wan+19; OWJ18; Llo82; JD88]). These offline approaches are not suitable for real-time large-scale scenarios, as they need to repeat the whole clustering process every time a new sample arrives. In addition, most of them have difficulties dealing with complex data distributions.

2.1.2 Results and contributions

To overcome the challenges previously mentioned, we present an online gaussian mixture-based clustering method (OGMC). Our proposal's key idea is that an identity may be represented by several distributions or clusters. Using feature vectors (f-vectors) extracted from the incoming faces, OGMC generates clusters that may be connected to others depending on their proximity and their robustness. Every time a cluster is updated with a new sample, its connections are also updated. With this approach, we reduce the dependency of the clustering process on the order and the size of the incoming data and we are able to deal with complex data distributions. The high-level idea of the method is exposed in Figure 2.2.

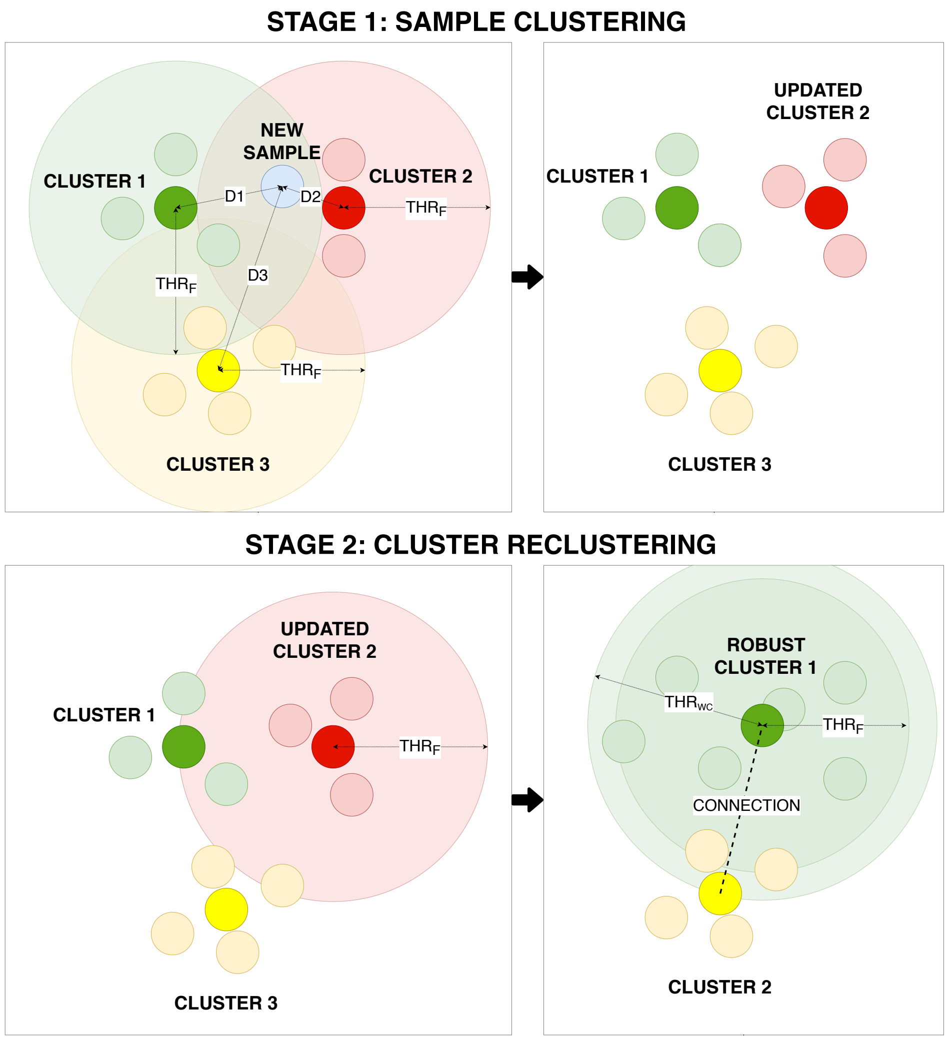


Fig. 2.2: Example of the operation of the proposed online clustering algorithm when a new sample arrives.

To demonstrate the potential of the proposed clustering algorithm, we conduct a series of experiments divided into two groups. The first group aims to measure the performance of OGMC in accuracy and processing time, comparing it with other traditional and state-of-the-art offline clustering methods. For that purpose, we test the model in three well-known face recognition datasets: IJB-B [Whi+17], IJB-C [Maz+18], and MS-Celeb-1M [Guo+16]. In each experiment, we use vectors extracted using a different face recognition model and of different sizes in order to demonstrate that the algorithm does not depend on them. Among all of them, the most challenging experiment is the one conducted using the MS-Celeb-1M dataset, where we use a face recognition model which extracts vectors with less features and a database with a much higher number of identities. The database contains 5.8M images from 86K identities. It is randomly split it into 10 parts with an almost equal number of identities. Then, 1 part is selected as labeled data for training and the other 9 parts as unlabeled data. With the unlabeled data, 5 tests are created with an increasing number of vectors and identities. The last test has 5.2M vectors and 77K identities. The results of this experiment are gathered in Table 2.1.

Tab. 2.1: Comparison with baseline methods in terms of BCubed F-Measure using subsets of different sizes from MS-Celeb-1M dataset. All methods use the same 256-dimensional vectors provided by [Yan+20b].

Method	Test				
	Number of samples				
	584K	1.74M	2.89M	4.05M	5.21M
	Number of identities				
	8.5K	25.7K	42.8K	60.0K	77.1K
	BCubed F-Measure				
K-means [Llo82; Scu10]	0.812	0.752	0.723	0.706	0.694
HAC [Sib73]	0.705	0.695	0.686	0.677	0.670
DBSCAN [Est+96]	0.672	0.665	0.663	0.449	0.447
ARO [OWJ18]	0.170	0.124	0.110	0.105	0.100
CDP [Zha+18]	0.787	0.758	0.746	0.736	0.729
GCN [Wan+19]	0.844	0.816	0.801	0.793	0.786
LTC [Yan+19]	0.855	0.830	0.811	0.798	0.789
GCN-V [Yan+20b]	0.858	0.826	0.811	0.799	0.791
GCN-(V+E) [Yan+20b]	0.861	0.828	0.812	0.801	0.793
OGMC (ours)	**0.906**	**0.881**	**0.864**	**0.851**	**0.839**

The second group of experiments focuses on testing its scalability, measuring the drop in accuracy and the increase in processing time as the number of data samples grows. In this case, the most challenging experiment conducted consists of testing the model using IJB-C vectors with 3M of distractors and measuring how the processing time per sample evolves with the number of processed samples and with the number of clusters in the database (note that the number of clusters is not equal to the number of identities, as we are not taking connections into account). Figure 2.3 shows that, after processing 3 million samples and with a database of 80 thousand clusters, the processing time per sample is still 5 milliseconds, which still allows OGMC to process 200 samples per second in real time.

Furthermore, an ablation study is presented in order to validate the design decisions, analyze the contributions of the different parts of the algorithm and measure the degree of dependency of the model parameters on the face recognition network and the train and test datasets.

Finally, to demonstrate the effectiveness of OGMC beyond face recognition, an additional experiment is conducted with DeepFashion [Liu+16], a dataset used for clothes retrieval. The results of the experiment are presented in Table 2.2. Among all

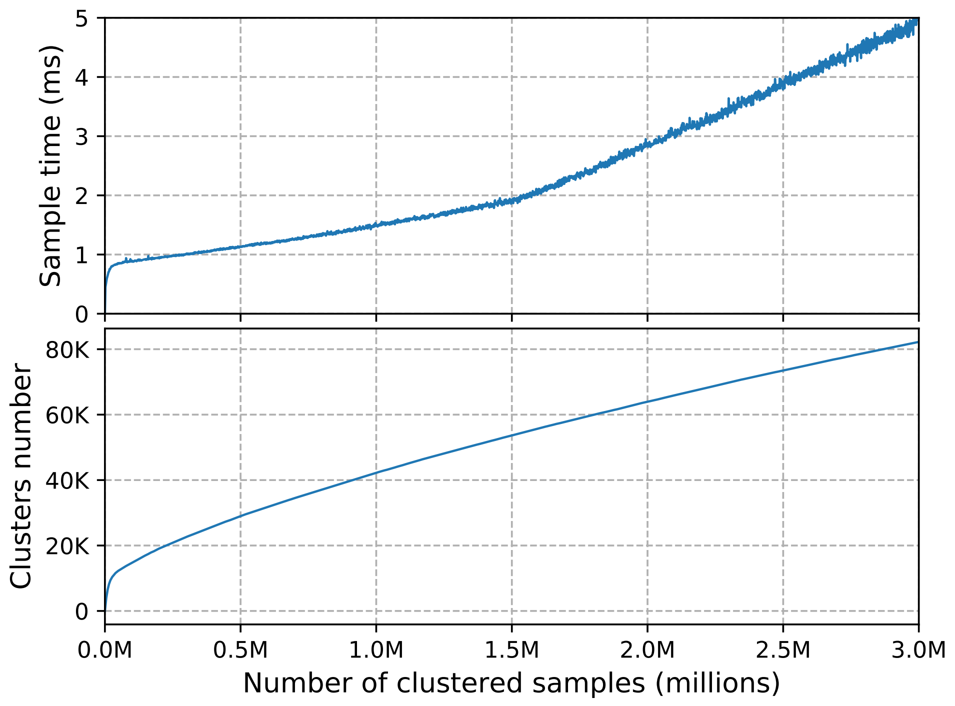


Fig. 2.3: Evolution of the clustering time per sample with the number of samples clusterized compared to the evolution of the number of clusters in the database.

the tested methods, OGMC achieves the best F-Measure, demonstrating its suitability for tasks beyond face recognition.

Tab. 2.2: Comparison with baseline methods in terms of BCubed F-Measure with DeepFashion dataset. All methods use the same vectors from [Yan+20b].

Method	F-Measure
K-Means [Llo82; Scu10]	0.538
HAC [Sib73]	0.488
DBSCAN [Est+96]	0.532
MeanShift [Yiz95; CM99]	0.567
Spectral [Ho+03; NJW01]	0.464
ARO [OWJ18]	0.530
CDP [Zha+18]	0.578
GCN [Wan+19]	0.589
LTC [Yan+19]	0.591
GCN-V [Yan+20b]	0.573
GCN-(V+E) [Yan+20b]	0.601
OGMC (ours)	**0.620**

From all these experiments, the following conclusions can be extracted:

- OGMC algorithm is independent of the recognition model: in all the cases the clustering algorithm improves the results achieved by its competitors.

- Despite being an online method, OGMC outperforms other online and offline methods in terms of F-Measure and processing time. For instance, in the IJB-B experiment, compared to the second best method (GCN-A [Wan+19]), OGMC achieves a better F-Measure while reducing the processing time by more than 6 times using the same hardware.

- The method is robust against outliers: in the results, we observe that the identity outliers are contained in non-robust clusters connected to the robust ones. This way, the quality of the robust cluster centroids is not compromised by these outliers and the number of matching errors is reduced.

- The 5 hyper-parameters of OGMC do not need to be readjusted if the scale of the dataset increases, so the user just needs to tune the parameters if the face recognition model is changed.

- The proposed method is suitable for real-time applications and highly scalable, even when dealing with extremely large amounts of data.

- OGMC is suitable for other tasks beyond face recognition.

Finally, this work contributes to the second research line defined in the thesis. It demonstrates the importance of the postprocessing algorithms (Obj2.4), which can highly improve the performance of the system in terms of accuracy and processing time. In this sense, it also contributes to the state of the art with a new online clustering method that outperforms the rest of the evaluated approaches.

2.2 Learning to Automatically Catch Potholes in Worldwide Road Scene Images

2.2.1 Motivation and objectives

This work tackles the challenge of pothole detection from images of real-world road scenes. It was developed in the context of the AUTOPILOT H2020 project [24], which aims to deploy, test and demonstrate automated services based on Internet of Things (IoT) in five driving modes. In the Highway Pilot use case, a cloud service merges the sensor's measurements from different IoT devices to locate and characterize road hazards. The goal is to provide the following vehicles with meaningful warnings and driving recommendations to manage the hazards in a safer or more pleasant way.

For a better understanding of the scenario, we assume the following: Firstly, a vehicle equipped with different systems has the role of IoT device, which is comparable to smartphones and other wearables that can send/receive messages to/from IoT platforms. Secondly, in-vehicle systems include AI modules that process data and produce low-bandwidth messages that are wirelessly sent to IoT platforms. Figure 2.4 shows an illustration about the AUTOPILOT usecase.

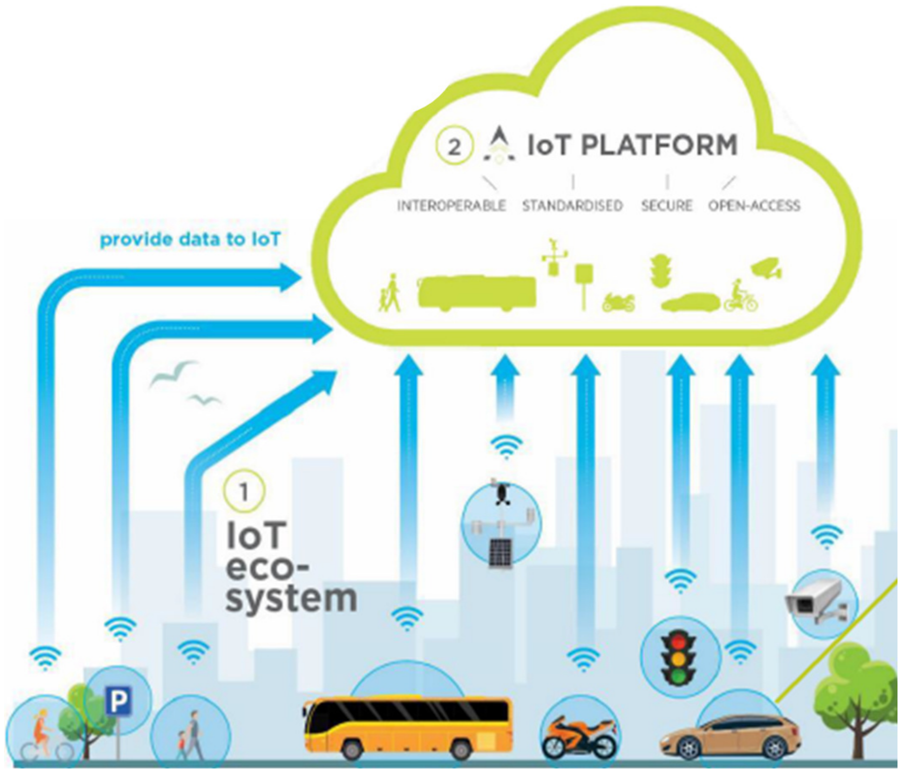


Fig. 2.4: AUTOPILOT H2020 project illustration [24]

Within this background, our main research motivation is the automated visual detection of road potholes from a frontal colour camera on board a vehicle. Once potholes are detected, their locations are reported to a given IoT platform. The current state of the art is predominantly based either on sensing potholes with accelerometers [Bha+17] or cameras [NBK15]. Accelerometer-based detection requires that the vehicle drives over the potholes, which is usually not the case as the driver will try to avoid them. On the other hand, vision-based detection is naturally seen as the same process in which drivers perceive the environment and anticipate to possible road hazards. For the latter one, classical image processing and machine learning approaches have been evaluated on images of certain world regions. Our strategy is to automatically learn the visual appearance of worldwide potholes using the latest advances in Deep Neural Networks.

2.2.2 Results and contributions

The first step for developing the model is collecting and preparing the training data. After searching through the literature, we found only one pothole dataset available [NBK15]. It consists of 4,030 images of size 3680×2760 pixels and their labels. The images were captured as regular snapshots from a GoPro camera attached to the inner side of a vehicle windscreen. In order to increase the size of the dataset, we sample 1,644 images from AUTOPILOT videos of VALEO in the surroundings of Paris and manually labelled the potholes. These images have a resolution of 1280×800 pixels. Finally, to add more diversity to the dataset, we capture 100 images from different locations using the Google Earth Pro street-view tool at a resolution of 1236×804 pixels. They have been also manually labelled. Thus, the database consists of 5,874 images from which 5,774 have been used for fine-tunning and 100 have been randomly picked for validation. In the training set there is a total amount of 9,716 potholes while in the validation set the number of potholes is 171.

After gathering all the data, we perform an exploratory data analysis (EDA). We have collected images with several types of potholes under different illumination and weather conditions as depicted in Fig. 2.5.

The most common potholes show a pronounced edge describing an elliptical shape. However, there are some that describe a more square-like shape and some others that do not have a pronounced edge. They might appear darker or brighter on the background pavement. The deepest ones look darker because of the shadow of the edge, while in the flat ones it is possible to see the ground or gravel inside the pothole. Besides, some potholes appear filled with water and might also reflect surrounding scene in the surface. We also extract information about the size of the images and the distributions of the size, aspect ratio and area of the annotated potholes, which is useful for the later design of the DNN. We provide box plots in Figure 2.6.

Regarding the DNN, based on the investigation in [Hua+17] and the entries reported in the TensorFlow model Zoo [14], we select 4 models that have shown high detection ratios at reasonable processing costs. Besides, we also evaluate the Single Shot multibox Detector (*SSD*) because it is targeted for mobile applications. Attending to detection performance we choose 3 different configurations of the architecture *Faster R-CNN*:

1. Faster R-CNN Inception v2. The feature extractor in this approach provides batch normalization for accelerating the model training and it has yielded high

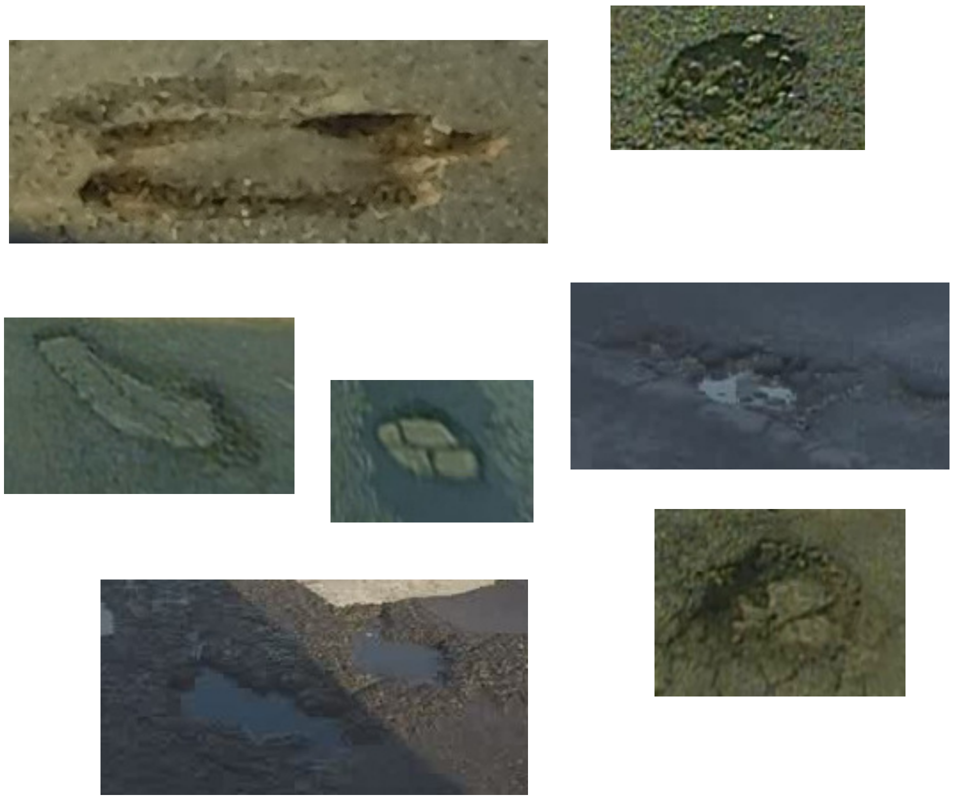


Fig. 2.5: Samples of annotated potholes. They come from several sources and represent varied places, environmental conditions and camera viewpoints.

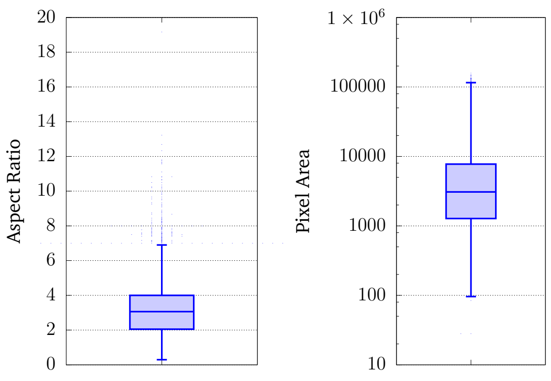


Fig. 2.6: Distribution of the aspect ratio and area in pixels of the annotated potholes for the train dataset. The boxplot on the left has a linear representation of the aspect ratios. The quartiles are $Q_1 = 2.053, median = 3.062, Q_3 = 4.0$. The boxplot on the right shows the area in pixels and has a logarithmic axis for the purpose of visualization. The quartiles are $Q_1 = 1276, median = 3081, Q_3 = 7744$ pixels

accuracy [How+17]. Compared to v1, it is more efficient by factorizing the convolution operations. Also, it is a wider network to avoid losing visual details that may happen in v1 which is deeper as road potholes typically represent small portions of the images.

2. Faster R-CNN Resnet101. This model training uses Resnet101 that stands for Residual Network with 101 layers [How+17] and has achieved high success on many competitions. This type of networks try to learn residuals which are short-cut connections between layers. The approach allows to train deeper models without degradation.

3. Faster R-CNN Inception-Resnet v2 (atrous). This third model uses a hybrid feature extractor that combines Inception and Resnet, second version (v2), yielding improved recognition performance as reported in [Sze+16]. Moreover, the "atrous" (*with holes in*) option employs dilated convolutions, which provide a wider field of view at the same computational cost towards achieving better accuracy.

Attending to the performed EDA, we fix the size of the input layer to 1024×800 pixels. We found that the median area of potholes corresponded to a 0.37% of the original image size. Given a reduced resolution of 600×600 as reported in [14], $1350 \simeq 36^2$ pixels is the average area of the potholes in the images. Besides, the 1:1 aspect ratio involves warping, cropping and resizing the original images. Consequently, road areas on the left and right sides in front of the car are cropped out and the visual appearance of the scene and the potholes are deformed and reduced in granularity. Therefore, we decide to use a larger window size with an aspect ratio similar to the original resolution of the images, achieving a median area of 3081 pixels among the annotated potholes.

In addition, we apply a set of adjustments in the architecture and the training hyper-parameters for the 3 Faster R-CNN models.

- Add the aspect ratios of 1:3 and 1:4 after analysing the dataset (see Fig. 2.6).

- Reduce the maximum number of region proposals for Faster R-CNN models from 300 to 100, with the aim of decreasing detection time without losing performance [Hua+17].

- Due to the limited size of our pothole database, we increase the number of training samples by data augmentation. Among several options for fine-tunning Faster R-CNN models, we select *random_adjust_brightness* and *random_horizontal_flip* because of their effectiveness in previous research experience for object detection.

- We enable the drop-out feature in the second stage of the Faster R-CNN training to prevent over-fitting. Basically, this option randomly drop units and their connections from the network, which prevents the units from co-adapting too much.

- The number of steps employed for fine-tuning the 3 models is 2M. We pick this number observing the performance of the trained models on the validation subset. The loss function was stable without showing too much improvement, thus we decide to stop at 2M steps, which corresponds to 173 epochs.

Similarly, we apply the same adjustments for fine-tuning the *SSD Mobilenet v2* model but using a a squared size of 800 $\times$ 800 pixels.

For the models evaluation, we select the *mean Average Precision* (mAP) as detection performance metric and the higher the mAP, the better.

We start by evaluating *SSD Mobilenet v2*. The AP was 33.62% and 45.57% @ IoU 0.5 and 0.4, respectively and took approximately 455ms per image when executed in Nvidia DrivePX2. These values show a very low detection performance despite the fine-tunning of some parameters. We investigated the main reasons and reached the conclusion that in SSD Mobilenet the discretization in bounding boxes and their separate analysis does not account for neighbouring pixels, which provide useful context information. Despite the initial expectations of this research to implement a mobile embedded Deep Neural Network, the SSD Mobilenet is not well suited to detect objects that strongly depend on the appearance of surrounding scene, i.e. potholes vs road appearance.

Regarding the Faster R-CNN models, Figure 2.7 presents the precision-recall curves at two different values of IoU and running times are shown in Table 2.3.

Tab. 2.3: Average inference time for the 3 pothole detectors and different gpu devices.

	Inference time (ms)	
Models	**Tesla P100M 16GB**	**DrivePX2 TegraA**
inception_v2	53.2	177.1
resnet101	94.2	432.7
inception_resnet_v2_atrous	172.1	732.9

As it can be observed from the resulting values, **Faster R-CNN Resnet101** yields the highest detection performance. It is not the slowest DNN but it requires on average 432.7ms per frame on the Nvidia DrivePX2. For the goals of detecting potholes and reporting them to an IoT platform (AUTOPILOT project), it is a valid time. There are not real-time requirements because detected potholes will be monitored from control centres and broadcasted to warn other road users when sufficient confidence

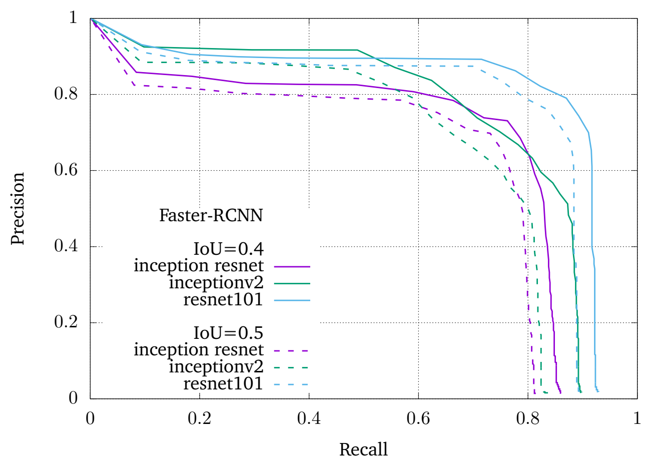


Fig. 2.7: Precision-recall curves for the test set. Three Faster R-CNN models with a different feature extractor network and two values of IoU are compared. Due to the nature of road potholes, there is a larger error in the localization accuracy of the ground-truth bounding boxes. Thus, we opted to report IoU = 0.4 in addition to the commonly used 0.5 value. The aim is reducing false negatives and false positives in road scenes with deteriorated surfaces where manual pothole labelling is also challenging.

is reached after several repeated detections on a given map location. For vehicle reactive manoeuvres upon detection, further research is needed to optimize the neural networks without losing too much performance in order to obtain processing gains.

To sum up, the contributions of this paper are the following:

- We have built a dataset with manual annotations from several places around the world that include images from Europe, America, Asia and Africa. It is composed of challenging scenes captured from different cameras, viewpoints and under varied environmental conditions. We have also performed and EDA to extract important information about the data.
- We have fine-tuned and evaluated 4 different Deep Neural Networks (DNNs) for the detection of road potholes on images. We have applied several modifications to the architectures and the training hyperparameters, as well as other techniques such as data augmentation or dropout. Consequently, we have achieved high detection ratios ($mAP > 75\%$) considering the high intraclass variance.
- The pothole detector has been tested on the Nvidia DrivePX2 platform for embedding ADAS in driverless vehicles.

- As part of AUTOPILOT project, the pothole detector has been integrated on a real vehicle. The set-up included an automotive grade camera, General-Purpose computing on Graphics Processing Units (GPGPU) and IoT communication modules on board the vehicle.

Finally, this works contributes to all the objectives from the first research line: the optimization of the DNN. It tackles the DNN selection (Obj1.1), the design of the whole training pipeline (Obj1.2), including data selection and preparation, and the optimization of the architecture (Obj1.3) with several modifications, like reducing the number of region proposals or modifying the aspect ratios.

2.3 Multi-Camera BEV Video-Surveillance System for Efficient Monitoring of Social Distancing

2.3.1 Motivation and objectives

In this work, we consider the problem of efficient monitoring of a set of established security measures in large infrastructures using non-invasive technology. More specifically, we aim to create a video-surveillance system capable of monitoring compliance with social distance and capacity limitation measures, as well as tracking the offenders or the possibly infected subjects. Therefore, the system must be capable of merging the information coming from multiple cameras to track subjects all over the monitored region. We use overlapped camera views to track people across views.

For this task, the system will extract the necessary information from a set of cameras placed on the ceiling of the monitored infrastructure. The number of cameras depends on the area to be covered and on the height of the ceiling. The setup needs to guarantee that all the areas of interest are visible by the cameras with a minimum resolution (limited by the capabilities of the human detection network). The omnidirectional cameras with fish-eye lenses have a wide field of view, which makes them appropriate for monitoring large areas with a minimum number of sensors. Intrinsic and extrinsic parameters of all the cameras need to be available for an accurate distance measuring. An image illustrating the considered use case is presented in Fig. 2.8. The system must be able to report reliable and real-time information about the state of measures compliance using minimum processing requirements, preferably a single-GPU server.

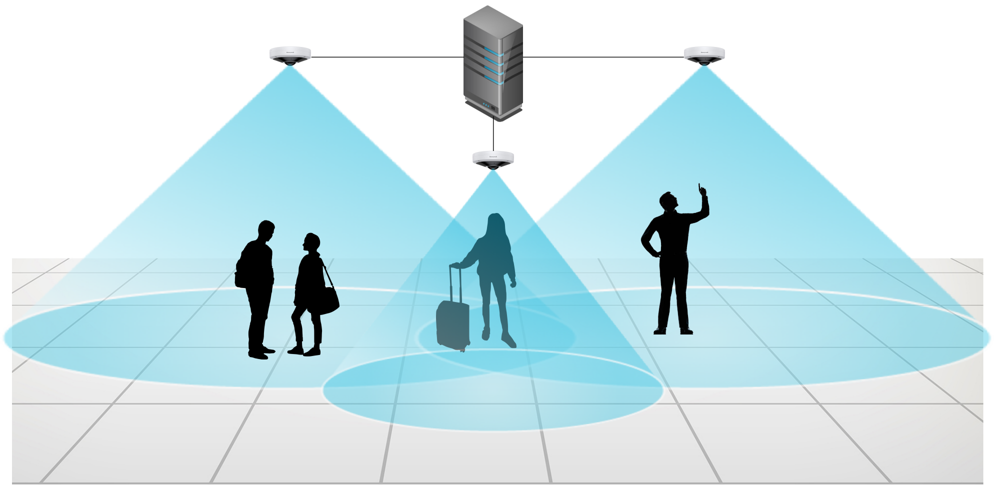


Fig. 2.8: Illustration of the considered scenario. Multiple omnidirectional cameras with a small overlap cover the monitored area. The cameras are connected to a central GPU-enabled server.

2.3.2 Results and contributions

Considering the requirements mentioned above, we carefully design the whole monitoring system. An overview diagram is illustrated in Figure 2.9 and a flowchart in Figure 2.10.

The system requires an initial camera calibration process in order to compute the intrinsic and extrinsic parameters. The estimated camera parameters are used in later stages for mapping the image coordinates to the 3D world coordinates and for locating each camera with respect to the others. First, the images are grabbed from each configured camera. Then, they are preprocessed in parallel using the CUDA library and fed into the pruned version of YOLO-V4 detector, implemented in TensorRT framework. Therefore, from the original (distorted) images, a set of detections is obtained for each camera at each frame. Each detection is modeled as a 4-point rectangle in image coordinates.

We alter the typical order of the tracking and projection stages for two reasons: it is easier to estimate the trajectory using world coordinates than image coordinates from an omnidirectional camera (where the bounding box varies rapidly); and this way we only need a single tracker instance to process all detections in world coordinates. Thus, from the detections, a fitting process yields the desired 3D cylinder shapes, where each cylinder is encoded using the center point in the XY plane, the height, and the radius. Then, a fusion mechanism determines which cylinders correspond to the same object for cameras with overlapped fields of view. Next, the tracking stage takes these cylinders, applying a constant-velocity predicting

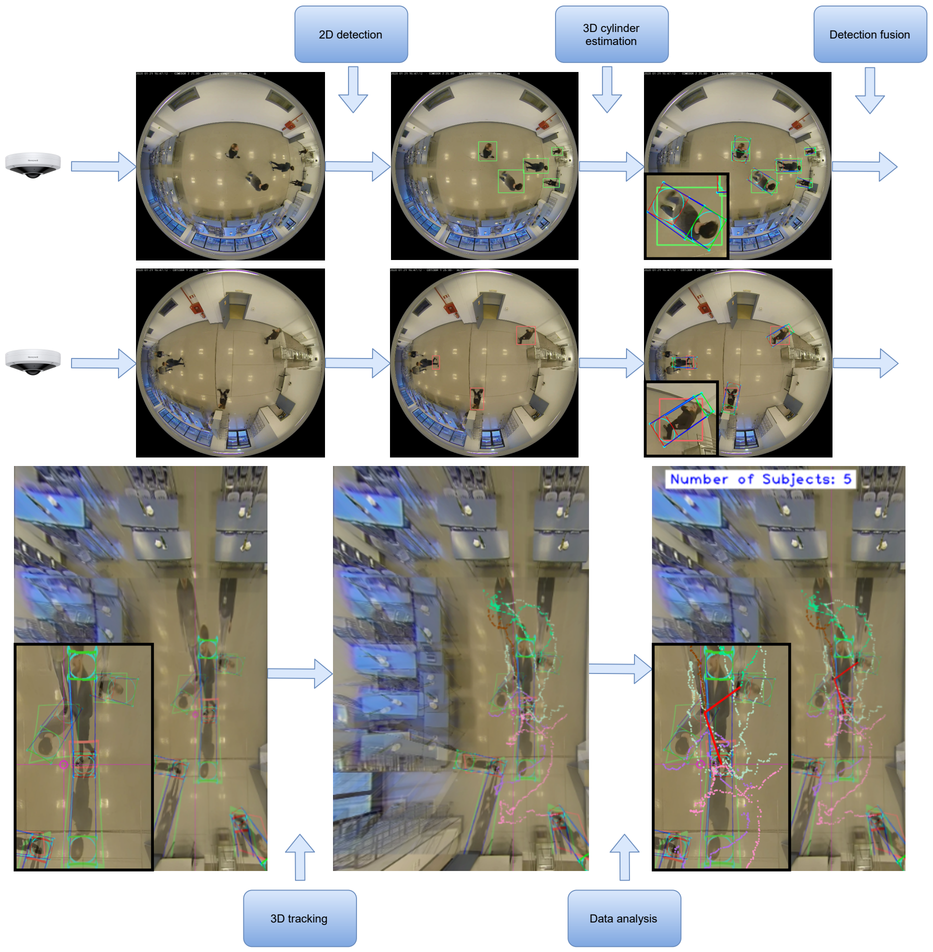


Fig. 2.9: Overview graphical diagram of the proposed workflow. Note that the rectified images (in the lower row) are generated only for visualization purposes and are not necessary for fusing the detections, tracking, or data analysis.

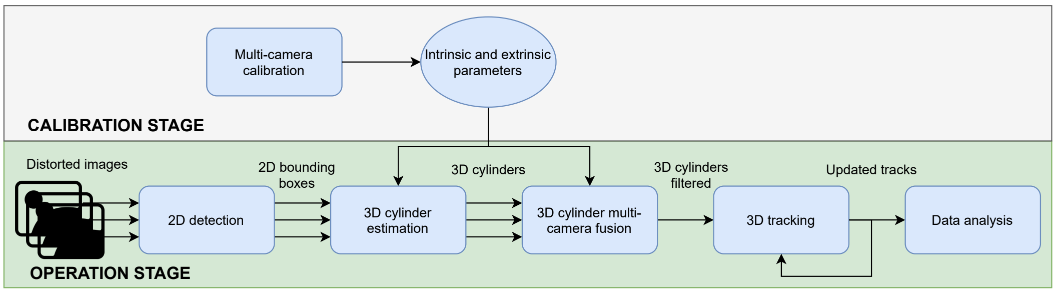


Fig. 2.10: Overview flowchart of the proposed workflow. Note that until the cylinders fusion step, the detections of each camera are processed separately.

model, plus managing the appearance and disappearance of objects, miss-detections, etc. Finally, the output time-consistent tracks are analysed to extract the necessary information for the monitoring of the established security measures.

About the object detector, Similar to [Ara+21], we train YOLO-V4 [BWL20a] to detect people directly in overhead images from fish-eye cameras. We use this single-stage detector because it provides a good balance between accuracy and inference time. Compared to previous versions, YOLO-V4 includes detections at three scales, which improves the small object detection accuracy.

To further increase the performance of the model we apply two optimization processes. First, we apply the weight pruning procedure described in [ZZL19]. Finally, the pruned models are ported to TensorRT framework to apply hardware-level optimizations. With these optimizations we are able to reduce the inference time by almost a 90% for each model, from 22 fps to 110 fps.

Regarding the training data, as there is no public dataset with top-view fish-eye images of large spaces focused on human detection and multi-camera systems, we use several recordings to build our training dataset. We set up two omnidirectional cameras installed at 3.3 meters and another camera at 8 meters. We capture 10,000 images for the lower height range and 10,000 images for the upper one. In addition, to augment both ranges' data variety we add 5,600 synthetic images from the Advanced Synthetic Dataset presented in [Ara+21] to each of the datasets.

We also apply data augmentation techniques to enrich the data. As shown in [Zop+20], rotation and histogram equalization are some of the most efficient image augmentations for training accurate object detection CNNs. Consequently, we apply rotations, flipping and histogram equalization augmentations (CLAHE) to our images. We randomly combine these augmentations and generate 4 new samples for each image.

We conduct a series of experiments to analyze the suitability of the proposed system. to evaluate the performance of the proposed system, we focus on measuring the quality of the tracks and the accuracy of the positioning. To measure the quality of the generated tracks, we use the metrics described in [Den+20], developed to precisely compare different multi-object tracking methods in crowded scenes. We generate 7 sequences with different scenarios, number of cameras, heights, number of identities, and levels of occlusion. The details of each sequence are presented in Table 2.4.

Furthermore, we also present an experiment comparing the proposed 3D cylinder estimation procedure with other alternative 3D projection methods:

- Projecting the center of the detection bounding box. This is the approach followed in other related works [Ahm+20; AAJ21; RA20; PSA20; Yan+20a].

- Projecting the point of the bounding box closest to the center of the image. For cameras with fish-eye lenses, the furthest point from the camera of a vertical

Tab. 2.4: Details of the different sequences considered for the system evaluation. For each sequence we specify the scenario, number of cameras, camera heights in meters, radius of the monitored area for each camera in meters, number of frames, number of identities, and occlusion level (from 1 to 5).

Seq	Sc	NC	Heights	Rad	Frame	IDs	Occ
1	1	2	3.3, 3.3	3.5	1610	3	1
2	1	2	3.3, 3.3	3.5	654	5	2
3	1	2	3.3, 3.3	3.5	750	5	3
4	1	2	3.3, 3.3	3.5	1083	4	4
5	1	2	3.3, 3.3	3.5	837	5	5
6	2	1	5.5	8	334	6	2
7	3	1	8.1	10	578	13	4

object (i.e. the feet position of a person) corresponds to the object point closest to the center of the generated image.

- Estimating the 3D cylinder. This is the proposed approach.

The results of the experiments are presented in Table 2.5. In this table, we compare the metrics for measuring the positioning accuracy: F1 score ($F1$) and average positioning error (APE). Note that the proposed approach achieves the best result by far, reducing the average positioning error by more than 4 times in most cases.

Finally, the contributions of this work are the following:

- A modification in the traditional tracking-systems pipeline that allows our system to operate efficiently in multi-camera environments. Unlike the rest of the proposed methods, we move the projection step to real-world coordinates just after the detection step for each camera (instead of applying it after the tracking step). Then, thanks to an initial multi-camera calibration procedure, we are able to track the subjects uninterruptedly all over the monitored area using just a single tracker instance for processing all the detections. Furthermore, this approach allows using cameras installed at different heights, since it does not take into account the detection bounding box for the multi-camera fusion but the real position in meters.

- A 3D projection and multi-camera fusion procedure. Using the intrinsic and extrinsic parameters of the involved cameras, it estimates the best-fitting 3D cylinder for each detected bounding box and fuses the cylinders of the overlapping regions of the camera views that belong to the same person. This

Tab. 2.5: Comparison of the performance of the proposed method using different 3D projection approaches: projecting the center of the bounding box; projecting the point closest to the center of the image; and estimating the 3D cylinder (proposed approach). S stands for the sequence number and C for the camera IDs involved in the test.

S	C	BBox Center		Closest Point		3D Cylinder	
		F1	APE	F1	APE	F1	APE
1	1,2	0.776	0.600	0.835	0.216	**1.000**	**0.094**
1	1	0.738	0.612	1.000	0.215	**1.000**	**0.089**
1	2	0.791	0.453	1.000	0.273	**1.000**	**0.069**
2	1,2	0.776	0.572	0.898	0.388	**0.997**	**0.151**
2	1	0.540	0.618	1.000	0.251	**1.000**	**0.071**
2	2	0.589	0.526	**1.000**	0.355	0.997	**0.063**
3	1,2	0.734	0.481	0.900	0.301	**0.987**	**0.136**
3	1	0.617	0.524	0.985	0.287	**0.991**	**0.084**
3	2	0.736	0.481	0.985	0.326	**0.986**	**0.077**
4	1,2	0.754	0.471	0.882	0.297	**0.986**	**0.155**
4	1	0.734	0.493	0.994	0.268	**0.999**	**0.085**
4	2	0.643	0.455	0.982	0.304	**0.993**	**0.095**
5	1,2	0.807	0.504	0.912	0.395	**0.991**	**0.186**
5	1	0.647	0.569	0.971	0.258	**0.978**	**0.097**
5	2	0.606	0.521	0.993	0.330	**0.996**	**0.085**
6	1	0.867	0.331	0.976	0.332	**0.991**	**0.089**
7	1	0.907	0.474	0.967	0.272	**0.971**	**0.112**

corrects possible occlusion problems and allows us to expand the useful range of the cameras.

- A full pipeline for training and optimizing an object detection DNN model for this use case. We have covered all the stages since the data selection and preparation, which includes generating additional synthetic data and applying data augmentation. We have selected a suitable architecture, trained it, and optimized it, achieving an increase of performance from 22 fps to 110 fps.

This work contributes to both research lines, as we have designed a whole pipeline for training (Obj1.2) and optimizing (Obj1.3) the DNN and we are improving the performance of the DNN-based system through the modifications in the pipeline and the proposed 3D projection and multi-camera fusion procedure (Obj2.4).

2.4 Multi-Stage Dynamic Batching and On-Demand I-Vector Clustering for Cost-effective Video Surveillance

2.4.1 Motivation and objectives

This work aims to design and build a cost-effective Video-Surveillance System (VSS) for face recognition and online clustering of unknown individuals at large scale. The main task of this system is to obtain reliable Performance Indicators (PIs) for people flow monitoring in large infrastructures, without storing any biometric information.

More specifically, our goal is to build a face recognition-based solution from which all the PIs can be derived for the whole infrastructure, trying to simplify as much as possible the required hardware setup, and with a better handling of data so that privacy issues can be avoided. This requires building an accurate and efficient face recognition system that can manage large-scale data, without storing the biometric information of the individuals.

For this purpose, we focus on how to take advantage of a central GPU-enabled computing server, connected to a set of video-surveillance cameras, to automatically register new identities and update their descriptive data as they are re-identified.

Thus, the objectives of this work are the following:

- Designing the whole video-surveillance system. It must be fast, efficient, and scalable, capable of working in real-time and of extracting reliable PIs.
- Selecting, training, and optimizing the different DNNs required for detecting and recognizing faces.
- Developing an online and extremely light clustering algorithm to re-identify the subjects in real-time.

2.4.2 Results and contributions

Figure 2.11 illustrates the overall processing architecture and data flow diagram of the proposed VSS. It is divided into three computing threads (the first for image capturing, the second for image processing, and the third for identity clustering) that are executed asynchronously, using CPU and/or GPU capabilities depending on the case, and sharing data sequentially.

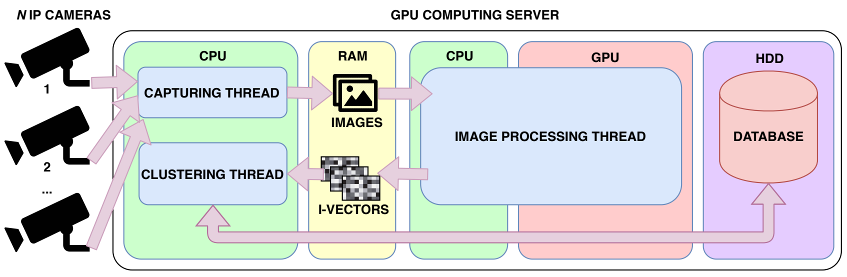


Fig. 2.11: Overall processing architecture and data flow diagram of the proposed VSS.

The capturing thread is in charge of three tasks: camera connection handling, stream decoding, and image acquisition. The computational cost of the capturing thread is low compared with the rest and it only uses a small amount of the CPU for frame decoding, mask applying, and connection checking.

The image processing thread applies the DNN-based face analysis algorithms to the captured images, in order to get the required identity vectors (i-vectors) for the re-identification. It consumes most of the computation time of the entire VSS (90-95%). Thus, to improve the global performance, the system exploits the GPU resources for the DNN inferences, using the CPU for minor tasks like processing flow control, image cropping, and managing patch lists (Figure 2.12).

The first stage detects all the faces present in the input image list and their facial landmarks, which are used for the face patch normalization following the method described in [Wan+18]. For an efficient detection of the facial regions and landmarks in the N images from the capture process we propose a multi-batch version of MTCNN (MB-MTCNN), with a batch size equal to the number of images (i.e. N). This stage generates a list of M face regions with a set of five facial landmarks.

In the proposed MB-MTCNN, the key factor to accelerate this process is the inclusion of parallel while loops, built upon flexible and expressive control-flow primitive operators [Ten17], executed in *execution frames* of the GPU that can be nested, allowing further optimizations. Thus, in the first parallel while loop of MB-MTCNN, the images are scaled and processed in batch using P-Net to obtain the region candidates. With a nested parallel while loop, the candidates of each scale and

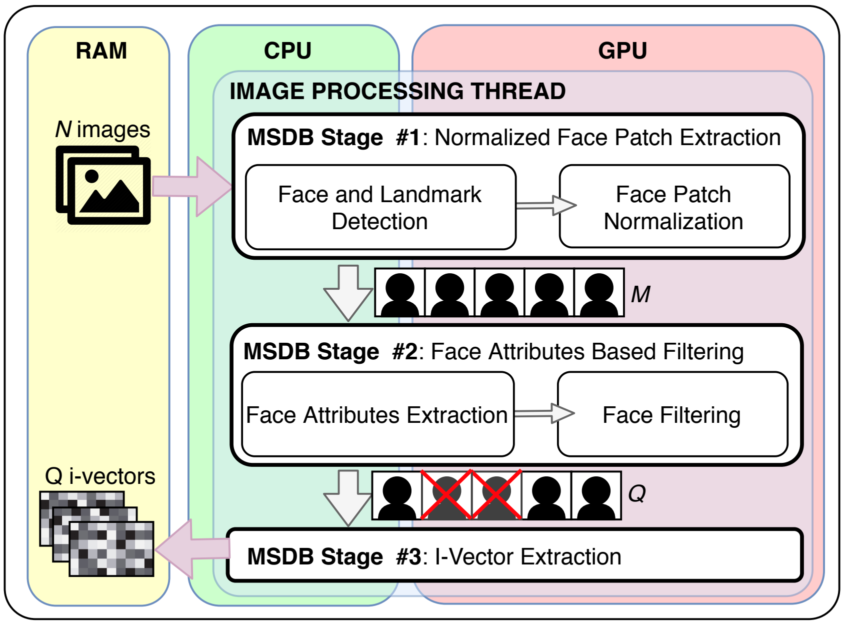

RAM
CPU
GPU
IMAGE PROCESSING THREAD
N images
MSDB Stage #1: Normalized Face Patch Extraction
Face and Landmark Detection
Face Patch Normalization
M
MSDB Stage #2: Face Attributes Based Filtering
Face Attributes Extraction
Face Filtering
Q i-vectors
Q
MSDB Stage #3: I-Vector Extraction

Fig. 2.12: MSDB procedure in the image processing thread.

image are postprocessed following the original implementation; scaled to their real size, refined and filtered with a non-maximum suppression (NMS). Finally, the candidates from all the scales are grouped by image in batch and NMS is applied again to merge candidates of different scales using a parallel while loop. In the subsequent stages, the candidates are filtered and refined using R-Net and O-Net networks. In these stages, the batches of candidates from each image are processed in parallel. This is more efficient than processing all the candidates in one batch, as it avoids reallocating the candidates for preprocessing and postprocessing.

The second stage checks if the facial patch is suitable for re-identification using a set of automatically detected attributes and previously defined filtering considerations. In our context, we use head orientation and a set of angle thresholds for the filtering process, but other descriptive attributes could also be considered (e.g. age, gender, ethnic group, etc.). These attributes are estimated with Multi-Task Cascaded Convolutional Networks (TCDCN) [Zha+16b]. All the patches that do not match the established rules are removed.

Finally, the last stage extracts the i-vectors from the list of filtered face patches given by the filtering stage. For our experiments we use a DNN model based on ResNet100 architecture [He+16a] with ArcFace loss [Den+19a]. We use this architecture, despite its complexity, because we need to generate the i-vectors as robust as possible in order to get highly-reliable PIs. The DNN inferences applied to the dynamic lists of cropped facial images are made following efficient dynamic batching procedures.

More specifically, our dynamic batching approach for stages 2 and 3 of MSDB is to load and infer multiple instances of the network, optimized for different batch sizes. Thus, for each inference, we divide the input batch size in the minimum number of mini-batches that fit the different network sizes.

Parallel to the processing thread, the clustering thread analyzes the incoming i-vectors to automatically register new people and update their descriptive data with new samples as they are re-identified, in an unsupervised way. In a summarized way, each new i-vector is compared with the centroids of the registered identities using the cosine similarity distance and a threshold. This threshold varies depending on the number of members of a cluster (how robust it is). If there are suitable cluster candidates, the i-vector is merged with the most suitable one and the centroid is updated. Otherwise, it generates a new cluster.

We conduct experiments to evaluate the performance of the different system components and the overall system performance. First, we test the clustering algorithm using two well-known datasets: IJB-B, using the i-vectors extracted in [Wan+19]. and IJB-C, using our face recognition model to extract the i-vectors. The results presented in Tables 2.6 and 2.7 show that the proposed method highly outperforms the others in the processing time and that this difference increases with the size of the database, which demonstrates that it is also more scalable. In terms of accuracy, the model achieves the first position in the IJB-C test, but the third position with the IJB-B dataset, where the quality of the i-vectors is worse. This shows that this method is more sensitive than others to the quality of the i-vectors.

Tab. 2.6: Comparison with baseline methods in terms of BCubed F-Measure and processing time using IJB-B-1845. Superscript* denotes results reported from original papers, otherwise, it uses the i-vectors from [Wan+19]. Times are in hh:mm:ss format.

Method	F-Meas	Time
ARO [OWJ18]	0.755	00:01:13
PAHC* [Lin+18]	0.610	00:03:56
ConPaC* [SOJ18]	0.634	02:53:58
DDC [Lin+18]	0.800	00:05:32
GCN [Wan+19]	**0.814**	00:06:03
ODIVC (ours)	0.778	**00:00:28**

Then, we evaluate the performance of MB-MTCNN compared to the original implementation. Figure 2.13 shows the average time of the detection stage per image for different resolutions and batch sizes using our approach (MTCNN corresponds to batch=1). The results show a great reduction in the processing time per image (more than 5 times for 720p). This reduction increases with the batch size but reaches a saturation point due to hardware limitations.

Tab. 2.7: Comparison with baseline methods in terms of BCubed F-Measure and processing time using IJB-C dataset. All methods use the same i-vectors extracted from our VSS and the same hardware. Times are in hh:mm:ss format.

Method	F-Meas	Time
ARO [OWJ18]	0.768	00:09:39
GCN [Wan+19]	0.890	00:10:32
ODIVC (ours)	**0.931**	**00:00:52**

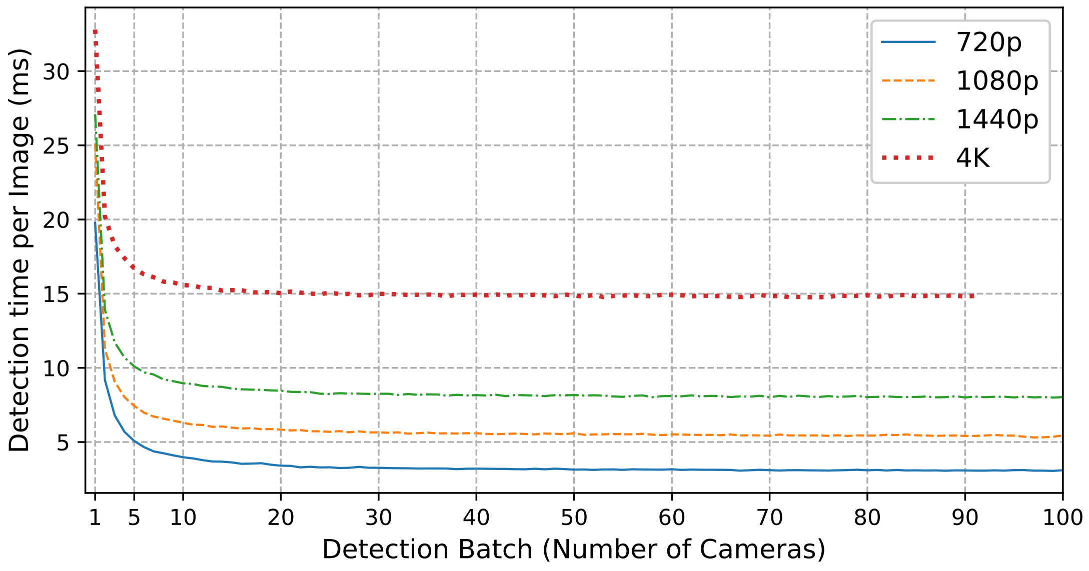


Fig. 2.13: Detection time per image for different resolutions and batch sizes in MB-MTCNN (in stage 1 of MSDB), compared to MTCNN (batch=1).

We also test the performance of the proposed dynamic batching procedure compared to other alternatives in the literature. We measure the recognition time per face when the VSS is processing a sequence of images containing variable numbers of faces. To better visualize time variations, we augment the number of faces by a factor of 10, so they may vary from 10 to approximately 1000. The results are shown in Figure 2.14, where TF stands for TensorFlow and TRT for TensorRT. The mini-batch size selected for the Parallel-Loop-TF approach is 20 and those used for Multi-Instance-TRT are 400, 200, 100, and 20. It can be observed that the Multi-Instance-TRT outperforms the other approaches, not only in the average but also in the maximum peak times.

Finally, to test the potential scalability of our VSS, we run it to process images captured from a scaling number of videos, at different resolutions and with a detection batch size set with the same value as the number of videos. Then, we measure the average times per image batch in the image processing thread, the main bottleneck of the system. The results are shown in Figure 2.15.

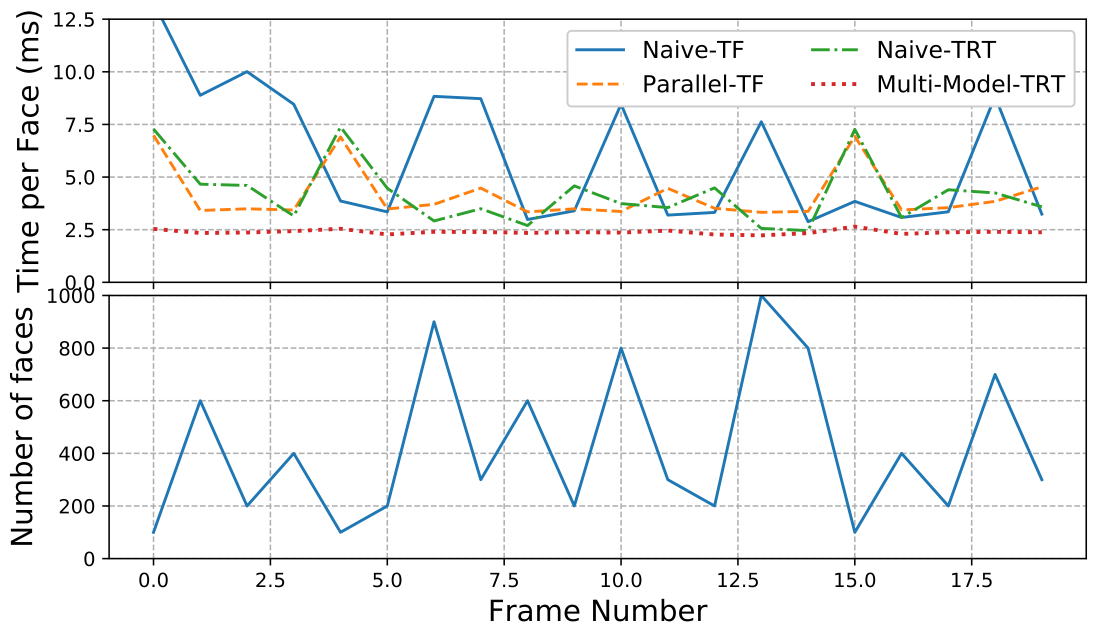


Fig. 2.14: Average time per face of the dynamic batching procedure for stages 2 and 3 of MSDB, compared to alternative state-of-the-art approaches.

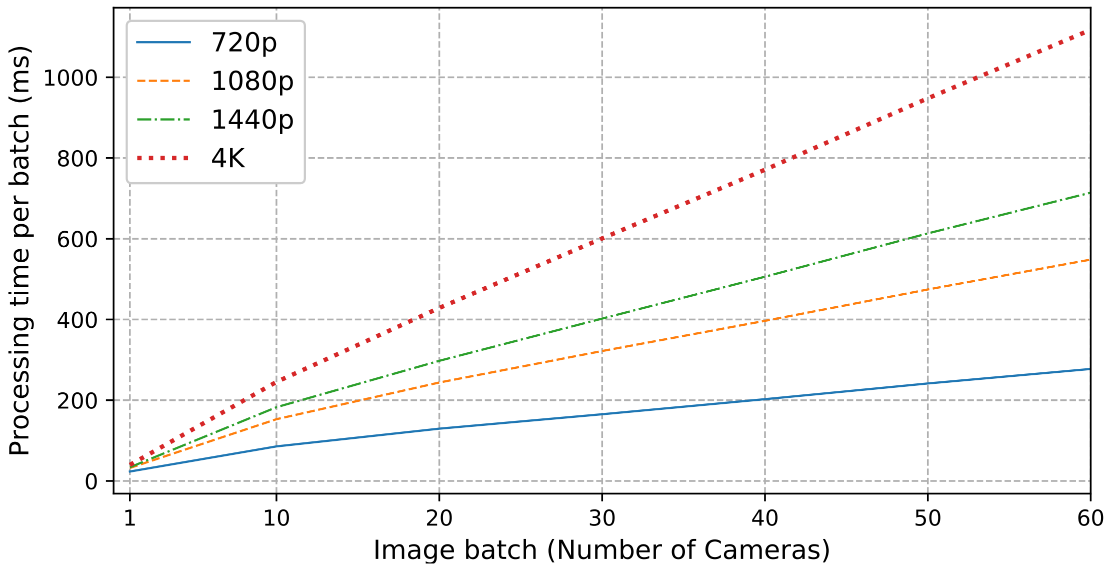


Fig. 2.15: Average times per batch (for the image processing thread) with the considered setup for different resolutions.

In our context, it is enough to deliver the PIs with near real-time performance. Hence, if we consider it acceptable that the processing thread responds every 200 ms, this setup could theoretically be scaled up to 40 720p cameras. The results reveal that the proposed VSS is suitable for designing cost-effective GPU-server-based solutions for our purpose.

To sum up, the contributions of this work are the following:

- A cost-effective VSS for face recognition and online clustering of unknown individuals at large scale, without storing their biometric information, to obtain PIs for people flow monitoring.

- A Multi-Stage Dynamic Batching (MSDB) procedure to efficiently extract face attributes and i-vectors from captured images. This includes MB-MTCNN, a multi-batch version of a Multi-Task Cascaded Convolutional Network (MTCNN) [Zha+16a], and an efficient dynamic batching strategy for the processing of dynamic lists of facial images.

- An On-Demand I-Vector Clustering (ODIVC) algorithm, designed to progressively adapt to the data scale, with a lower decrease in its effectiveness compared to state-of-the-art alternatives.

Attending to the objectives of this thesis, this work contributes to both research lines:

- **DNN optimization**. We have optimized the face detection model MTCNN to highly boost its performance (Obj1.3). With all the applied parallelizations, we have achieved a great reduction in the processing time per image (more than 5 times for 720p resolution).

- **CV System optimization**. We have carefully designed all the components of the system to maximize its performance:

  - Efficient capturing and preparation of the images in a separate thread to reduce the latency (Obj2.1).

  - A Multi-Stage Dynamic Batching (MSDB) procedure to efficiently extract face attributes and i-vectors from captured images for an optimal use of the hardware resources (Obj2.3).

  - An online clustering algorithm, extremely light, fast, and scalable, while achieving competitive results compared to the offline alternatives (Obj2.4).

2.5 Boosting Masked Face Recognition with Multi-Task ArcFace

2.5.1 Motivation and objectives

In this work, we address the problem of face recognition with masks. Given the global health crisis caused by COVID-19, mouth and nose-covering masks have become an essential everyday-clothing-accessory. This sanitary measure has put the state-of-the-art face recognition models on the ropes since they have not been designed to work with masked faces. In addition, the need has arisen for applications capable of detecting whether the subjects are wearing masks to control the spread of the virus.

Thus, we consider the problem of facial recognition of subjects who may or may not wear masks. As we do not know if the subject is wearing a mask, the network must perform well in both cases.

To solve this problem, we aim at increasing the accuracy of the face recognition network when dealing with masked faces, while preserving as much as possible the original accuracy with non-masked faces. In order to achieve this, the network must learn if the subject is wearing a mask to decide which facial features can be trusted in each case.

2.5.2 Results and contributions

We propose an approach based on the ArcFace work presented by Deng et al. [Den+19b] with several modifications for the backbone and the loss function. From the original face-recognition dataset, we generate a masked version using data augmentation, and we combine both datasets during the training process. We modify the selected network, based on ResNet-50 [He+16b; He+16a], to also output the probability that a face is wearing a mask without adding any additional computational cost. Furthermore, we combine the ArcFace loss with the mask-usage classification loss, resulting in a new function named Multi-Task ArcFace (MTArcFace).

An illustration of the proposed training pipeline is shown in Figure 2.16.

For the generation of the masked version of the dataset, we use the tool MaskTheFace [AR20]. The types of masks considered are surgical, surgical green, surgical blue, N95, cloth and KN95. The type mask is selected randomly and there is a probability of 50% of applying a random color and a probability of 50% of applying a random texture. Some examples of the generated faces are shown in Figure 2.17.

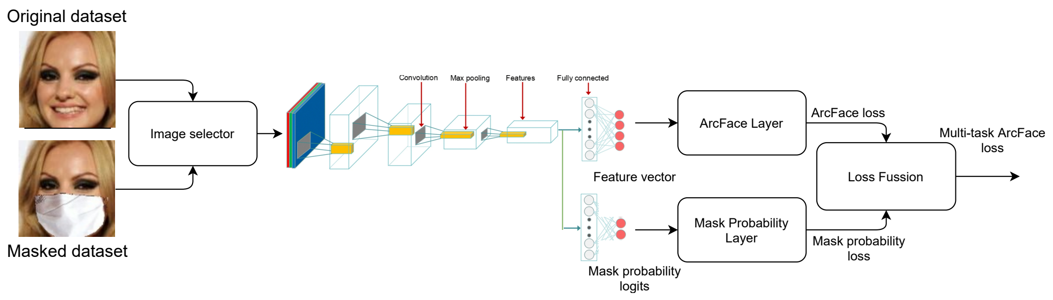


Fig. 2.16: Illustration of the proposed training pipeline. The image selector decides whether the next input image should be masked or not. The trained network is modified to output also the probability that the face is wearing a mask.

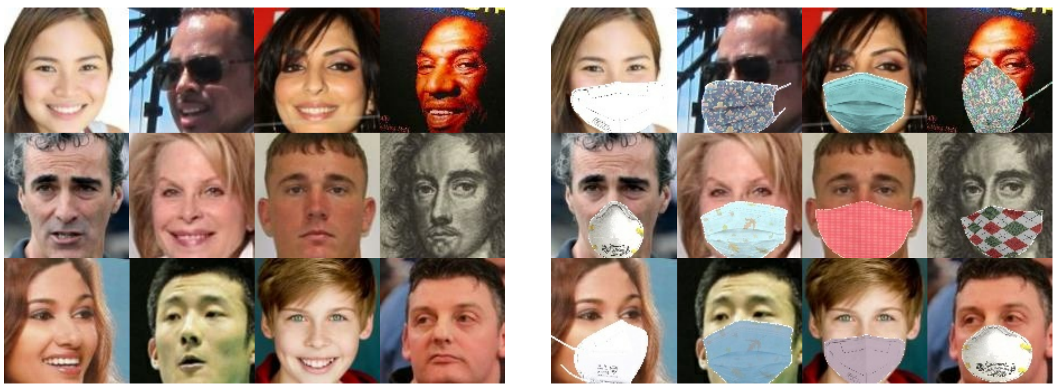


Fig. 2.17: Some examples of the training faces and their corresponding masked version generated with the MaskTheFace tool

.

Both datasets are shuffled separately using the same seed and, for every new face image selected for the input batch, we decide whether the image is taken from the original or the masked dataset with a probability of 50%.

We select LResNet-50 as the backbone among all the network architectures tested in the ArcFace repository as it is the one with the best trade-off between the accuracy and the number of parameters. More specifically, we use our own implementation of the network in TensorFlow DL framework, publicly available in a GitHub repository [Mon19].

Starting from this network, we add another dense layer parallel to the one used to generate the feature vector, just after the dropout layer, as shown in Figure 2.16. The new dense layer generates an output with two floats, which correspond to the scores related to the probability that the face is masked or not, respectively. This way, we force the network to learn when a face is wearing a mask, information that will also be used by the layer that generates the feature vector.

To extract the combined error from both logits, we start by generating the ArcFace loss ($loss_{ArcFace}$) in the same way as in the original paper. Next, we calculate the loss associated with the probability of wearing a mask ($loss_{Mask}$) by applying the softmax activation function on the logits and cross-entropy with the labels:

$$loss_{Mask} = crossEnt(Softmax(logits_{Mask}), labels_{ID}) \tag{2.1}$$

The Multi-Task ArcFace loss ($loss_{MTArcFace}$) is obtained by adding these two losses. However, to reduce the impact of $loss_{Mask}$ and give more importance to the ArcFace loss, we use the logarithm of $loss_{Mask}$ instead of the original value:

$$loss_{MTArcFace} = loss_{ArcFace} + log(loss_{Mask} + 1.0) \tag{2.2}$$

Finally, we add the regularization loss (as in the original implementation) to compute the total loss that will be used for the optimization:

$$loss_{total} = loss_{MTArcFace} + loss_{regularization} \tag{2.3}$$

We present the results of a series of experiments aimed at demonstrating the capabilities of the proposed method. We divide the experiments into two groups: identity verification and mask-usage verification.

For the verification task, we generate masked versions of 3 well-known face recognition datasets, also used in [Den+19b] for evaluating the original models. In addition, we also consider for the experiment the masked face dataset MFR2 described in [AR20], with 269 real-world face images from 53 celebrities, where the 64% of the faces wear a mask.

The results of the experiment are presented in Table 2.8. It can be observed that the proposed method largely outperforms the original model in the face verification task when dealing with masked faces. This increase in performance is more evident with profile images, where the amount of information of the face available is reduced, as is the case with CFP_FP, where the proposed model is almost a 12% more accurate than the original.

We also test the accuracy of the new model when recognizing non-masked faces, to check whether it has been a significant drop of performance. Thus, we repeat the previous experiment with the original non-masked datasets and compare the results with those achieved by the original model. The results, exposed in Table 2.9, Tab. 2.8: Comparison of the verification performance (%) with the masked datasets between the proposed method and the original ArcFace model.

Dataset	Proposed Method	Original model
Masked LFW	98.92	94.75
Masked CFP_FF	98.33	92.73
Masked CFP_FP	88.43	76.81
Masked AGEDB_30	93.17	90.53
MFR2	99.41	97.17

show that there is indeed a drop of performance for the new model, but that it is not significant (less than a 2% in the worst case). Furthermore, this drop in performance is much less than the gain obtained with masked faces. For example, in the case of CFP_FP, the model accuracy with masked faces increases almost a 12%, while its accuracy with non-masked faces decreases less than a 2%.

Tab. 2.9: Comparison of the verification performance (%) with the original datasets between the proposed method and the original ArcFace model.

Dataset	Proposed Method	Original model
LFW	99.45	99.62
CFP_FF	99.40	99.70
CFP_FP	92.27	93.81
AGEDB_30	95.02	96.90

Finally, we want to analyze the performance of the mask-usage probability output added to the proposed method. For this task, we run the model with all the faces contained in every masked and non-masked dataset used in the previous experiments. For each face we check whether the mask-usage probability estimated by the model is correct or not with a threshold of 0.5. Table 2.10 shows the results of the experiment. For each dataset, the model achieves nearly 100% accuracy. Again, the worst result is achieved for the CFP_FP dataset (98.82%) due to the profile faces. We believe that this is due to the fact that the training dataset does not contain enough profile faces. In any case, the model achieves an average accuracy of 99.78% across all datasets, so its effectiveness for this task is demonstrated.

This work contributes to the first research line (DNN optimization). We have presented a full-training pipeline for ArcFace-based face-recognition models to adapt them for working with masked faces (Obj1.2). This pipeline includes the generation of a synthetic masked dataset from the original training dataset. We have also modified the original architecture to output also the probability that the face is wearing a mask and we have designed a new loss function combining the outputs of

Tab. 2.10: Mask-Usage verification performance (%) of the proposed method.

Dataset	Accuracy
LFW	99.99
CFP_FF	99.99
CFP_FP	98.82
AGEDB_30	99.97
Masked LFW	99.98
Masked CFP_FF	99.98
Masked CFP_FP	99.70
Masked AGEDB_30	99.99

both tasks (Obj1.3). The results of the experiments show that the proposed method highly boosts the performance of the model when recognizing masked faces, while suffering just a small drop in performance with non-masked faces. Furthermore, they also demonstrate its effectiveness for the mask-usage verification task.

Finally, the proposed method and a modified version of it were selected to participate in the Masked Face Recognition Competitions (MFR) held within the 2021 International Joint Conference on Biometrics (IJCB 2021). Both methods were finalists and achieved second and fourth positions. All the details about the competition were published in [Bou+21].

2.6 BEV Object Tracking for LIDAR-based Ground Truth Generation

2.6.1 Motivation and objectives

In this work, we propose an offline 3D object tracking algorithm as a part of a semi-automatic annotation tool for 3D point clouds coming from LIDAR sensors.

Figure 2.18 illustrates the pipeline of the annotation tool which includes the proposed tracking component. The annotation process is divided into a series of steps, starting with the generation of the recordings from the sensorized vehicles.

LIDAR streams (3D point cloud sequences) are then pre-processed to create bird's-eye view (BEV) images (also called top view images), which are then used as input for the Convolutional Neural Network (CNN) detectors. The detector outputs are cuboids in 3D space, including the 3D position, size and rotation, the detection

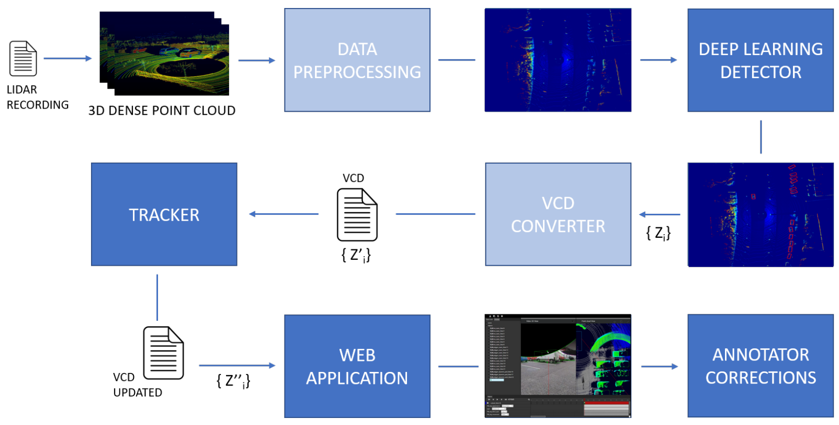


Fig. 2.18: Diagram of the annotation process.

class, and the confidence score. This data is then converted into a standard format using the VCD converter [NSO21] (a library developed by Vicomtech that offers tools for conversion between different formats), and will serve as the input data for the tracker. The tracker must be able to associate the input object detections between the different time steps, generate predictions where detections are missing and correct the input object properties by applying a post-process to the generated tracks. As a result, the VCD payload describing the scene is updated with the tracking information and sent to a web application for the final annotation refinement step carried out by human operators.

Thus, this works aims to develop an online tracking algorithm, robust to the detection errors and noise, in order to obtain the highest possible accuracy in the least possible time in order to work effectively within the semi-automatic annotation process.

2.6.2 Results and contributions

As mentioned above, the tracker is in charge of associating the input objects between the different time steps, generating missing tracking states in intermediate steps and correcting the input object properties by applying a post-process to the generated tracks. A diagram describing the proposed tracking process is presented in Figure 2.19.

The tracking process is divided into two main components. First, the online component is executed as a loop where, at each iteration, a new time step (i.e. frame) of the input data is processed. Tracks are then updated using predictions based on their previous information and the new input detections. In the correction step, an association matrix is generated and solved using a modified Branch and Bound Algorithm (BBA) [LW66] in order to match existing tracks predictions with detected

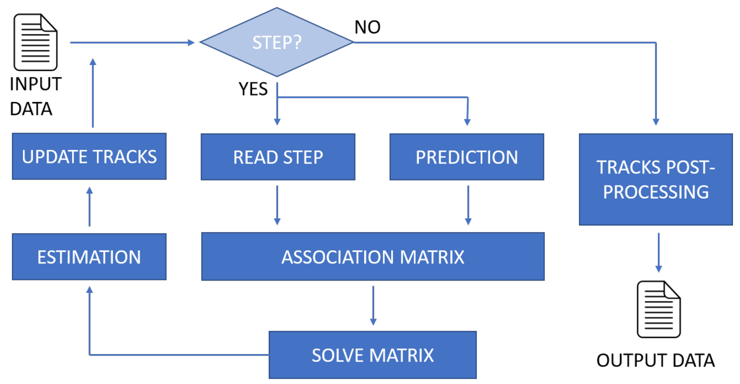


Fig. 2.19: Offline tracking dataflow where input data stands for the obtained CNN detections, and output data are the updated tracks.

objects and the tracks are updated. We also considered using the Greedy algorithm [Ann13] for solving the association matrix, as it is faster. However, analyzing the test results presented in Table 2.11 and the priority of the accuracy over the processing time it was discarded.

Tab. 2.11: Association algorithms comparison

Scenario	Steps	Errors	Max diff	GA time	BBA time
Street road	611	10	0.423	20 μs	35 μs
Parking 1	189	5	0.703	29 μs	45 μs
Parking 2	404	6	0.559	110 μs	132 μs

Finally, a post-process adds smoothness to produced tracks both in position and rotation. The aim of this stage is to remove orphan tracks and to correct the detection noisiest variables, the z-axis rotation angle, and the class. They are considered as orphan tracks all the tracks that have less than $\mathbf{n}$ states, and they will be removed since they have a high probability of being erroneous.

For the rotation correction, it is assumed that, in most cases, the detection value is right, and that it will change drastically from one step to another, so it will be calculated using the mode between $\mathbf{s} - \mathbf{n}$ and $\mathbf{s} + \mathbf{n}$ steps, being $\mathbf{s}$ the current step number. Also, the track size will be corrected using the mode of all the track states, as it should be constant in every step. Finally, for the track class variable, it is again assumed that, in most cases, the detection value is right, so it will receive the value of the mode between all track steps.

In order to validate the performance of the tracking algorithm we conduct a series of experiments. More specifically, we validate the tracker using the first 10 of the 22 training sequences with the ground truth of the KITTI dataset with the tracking metrics proposed in [Mil+16].

In the ideal situation when detections are perfect, using the ground truth boxes from the KITTI dataset, we have observed that the tracker produces perfect tracks, without error.

We also perform an ablation study on the input data in order to measure the impact of imperfect detections on the tracking results. We define three types of detection problems: (i) spatial noise; (ii) temporal sparsity; and (iii) a combination of both. The first test evaluates the effect of translation, rotation and size measurement noise in the detections. For this purpose, we define a probability of 50% to apply randomly a noise between -20% and 20% to each detection in each frame. The temporal sparsity refers to the miss-detections of objects in specific frames. In this case, we suppress 20% of each object detections randomly across the sequence. In all the tests, the tracker is able to overcome the noisy in many cases and achieves great results, including the case of the combined noise (see Table 2.12).

Tab. 2.12: Tracker results with combined noise

Seq	MOTA	F1	MT	PT	ML	FRAG
0	0.8770	0.9409	12.0	0.0	0.0	29.6
1	0.8778	0.9417	91.0	1.0	0.0	146.5
2	0.8935	0.9472	15.9	0.1	0.0	55.3
3	0.8454	0.9251	9.0	0.0	0.0	28.0
4	0.8735	0.9380	29.9	2.0	0.1	47.3
5	0.8640	0.9340	33.4	0.6	0.0	73.6
6	0.8693	0.9360	13.0	0.0	0.0	35.3
7	0.8689	0.9360	56.9	0.1	0.0	144.8
8	0.8771	0.9392	24.2	0.8	0.0	73.3
9	0.8783	0.9419	87.5	0.5	0.0	166.7

We finally analyze the results of the tracking when the input estimations are generated by the trained point cloud-based object detector. The same evaluation metrics are computed for this experiment. Results are shown in Table 2.13, where there is an extra column (F1D) with the F-Score obtained with the detector output. In

Tab. 2.13: Tracker results with real detector

Seq	MOTA	F1	F1D	MT	PT	ML	FRAG
0	0.5476	0.7650	0.7468	5	6	1	14
1	0.5389	0.7631	0.7420	50	20	21	92
2	0.3094	0.6491	0.6340	6	5	0	20
3	0.6265	0.8095	0.7800	4	2	3	9
4	0.6814	0.8490	0.7965	20	9	1	25
5	0.6280	0.8150	0.7077	14	9	11	34
6	0.7438	0.8788	0.8382	8	3	2	15
7	0.7000	0.8542	0.8473	43	11	3	70
8	0.5069	0.7554	0.7290	12	8	4	26
9	0.3406	0.6487	0.6382	29	26	27	106

this experiment, the MOTA results with real detections are, as expected, worse than with ground truth detections due to their inherent noise. However, there is always an improvement in F-Score, ranging between $2-5\%$ thanks to the usage of the tracker.

With this work we contribute to the second research line and we demonstrate again the importance of the optimization of the system components that interact with the DNN. We have designed the tracking algorithm taking into account the limitations of the DNN and trying to solve them as best as possible (Obj2.4), achieving improvements between $2-5\%$ in the accuracy. Furthermore, it also allows for reducing the processing framerate of the DNN, which is the main bottleneck of the system, as the tracker can fill the missing frames with the predictions.

Conclusions

3

3.1 Discussion

The main objective of this thesis is to provide knowledge and tools for the optimization of systems based on DL applied to different real use cases within the field of CV, in order to maximize their effectiveness and efficiency. To this end, at the beginning of the thesis, two research lines have been defined that seek to approach the problem from two different perspectives (see Section 1.4). The first one focuses on optimization of the Deep Neural Network (or networks) integrated within the system. This optimization ranges from the selection of the architecture to its optimization at the hardware level for deployment, including the selection and preparation of the data and the design of the training process. The second research line pursues the optimization of the system taking into account the rest of the components that interact with the DNNs, as well as its global design. This includes the design and optimization of the preprocessing, postprocessing, and transition components between networks (in those systems that have more than one).

During the years that the thesis has lasted, a series of research works dedicated to contributing to these lines of research have been conducted. Some of these works have given rise to the scientific publications on which this thesis is supported. Specifically, six main publications have been presented, and each one contributes to one or both research lines.

Regarding the first research line, the publications presented have evidenced the importance of the process of selection, training and optimization of DNNs to maximize their performance in each specific use case. Each detail of the process contributes towards achieving the optimal model. For instance, the importance of the architecture selection (Obj1.1) is evident in Section 2.2, where four different architectures are analyzed, three of them based on Faster R-CNN and the other on a Single Shot Detector (SSD). Contrary to what might be thought at first glance, it is shown that the architecture with a greater number of parameters is not the one that achieves the best results, and finally an intermediate model is selected, which achieves the best precision with a lower computational cost. This topic is also covered in Section 2.5, where several different architectures are studied and the one with the best trade-off between accuracy and processing time is found.

Another important part of the process that is emphasized in several publications is the selection and preparation of the training data (Obj1.2). In Section 2.2 the design of this step is key to achieving an accurate model. An in-depth study of the use case is carried out, as well as the object to be detected and, consequently, a rich and varied dataset is built that allows the objectives of the project to be achieved. In addition, an EDA is carried out, which allows extracting important information from the data, which is later used to optimize the architecture of the model. This step also plays a very important role in Section 2.3, where a specific dataset is manually built for the use case, capturing overhead images at different heights. Additionally, synthetic data is used to enrich the dataset. Finally, the selection of the data is also crucial in Section 2.5. In this case, a tool is used to synthetically generate a facial recognition dataset with masks. Combining this synthetic dataset with the original, a significant improvement is achieved in the reidentification of subjects with masks, maintaining the original accuracy of the model in uncovered faces.

Regarding the design of the training pipeline (Obj1.2), important parts of it are highlighted and multiple contributions are made throughout this thesis. For example, data augmentation techniques are applied to enrich the datasets in Sections 2.2 and 2.3. In addition, the selection of the optimizer, the loss function, and the optimal hyperparameters are also discussed in Sections 2.2, 2.3, 2.4 and 2.5. Finally, in all these sections, the appropriate metrics are selected and monitored in order to maximize the results for each specific use case.

The last important point that is addressed within this line of research is the optimization of the network architecture (Obj1.3). In this aspect we find multiple contributions in the publications. For example, in Section 2.2, small modifications to the architecture prior to training are introduced, such as the modification of the covered aspect ratios or the reduction of the number of region proposals for the models based on Faster R-CNN. Furthermore, in Section 2.3, post-training optimizations are applied to improve network performance. More specifically, the weight pruning technique is applied to reduce the number of network parameters and the model is ported to a framework optimized for inference (TensorRT). Modifications are also applied to improve network performance in Section 2.4, where the MTCNN face detection and alignment model is modified by parallelizing various parts of the network. As a result, an increase in fps of up to 5 times compared to the original model is achieved. Finally, in Section 2.5, the architecture of the facial recognition model is modified to allow it to perform a secondary task, classifying whether a subject is wearing a mask or not. Both tasks are complementary and contribute to the model learning faster and obtaining better results.

On the other hand, multiple contributions related to the second line of research have also been made. Above all, the publications highlight the importance of preprocessing and postprocessing algorithms (Obj2.2,Obj2.4) that interact with DNNs and the great impact that their design and optimization can have on system performance, in terms of precision, processing time and Resource consumption. For example, in Section 2.1 an online clustering algorithm designed to work as part of an application for real-time people re-identification is proposed. The algorithm is designed to work with feature vectors extracted from a facial recognition model based on DNNs. The experiments conducted on different datasets reveal that the proposed method is not only much faster than the rest, but also obtains a higher precision using the same feature vectors. Furthermore, this work shows also the importance of the data management components (Obj2.1) and of the transition components between the different DNNs included in the system (Obj2.3).

Another example about the contribution of a postprocessing algorithm to the performance of the system (Obj2.4) is presented in Section 2.4. Here, another online face clustering algorithm is presented along with the design of the complete system. Thanks to its minimalist design, it is capable of running in real time on a separate thread, at the same time as the rest of the system, but without sacrificing competitive accuracy. This work also shows the impact of the design of other important components such as data preprocessing, or the transition components between the different neural networks that the system includes.

A different example about the importance of the postprocessing algorithms (Obj2.4) can be found in Sections 2.3 and 2.6. In these works, the task requires using tracking algorithms to generate trajectories of people or vehicles detected using DNNs. In both cases, the algorithms are designed taking into account the use case and the peculiarities of the networks that precede them and allow to overcome some of their weaknesses and maximize system performance. In addition, in Section 2.3, modifications are also made at the global level in the traditional pipeline that is followed in the literature and a new projection algorithm from 2D to 3D detections is designed. These modifications improve the accuracy of the system and the proposed projection approach achieves better results compared to other state-of-the-art approaches.

All the optimizations carried out and the algorithms proposed in the research papers presented in this thesis demonstrate the need to invest time and resources in the optimization of systems based on Deep Learning at all levels to maximize their performance. Furthermore, all these procedures and algorithms can be adapted and applied to other use cases, not only within the field of Computer Vision, but also in the field of Deep Learning in general.

3.2 Future work

As future work, there are still many optimization techniques that have not been analyzed and applied in this thesis, some because they were not appropriate for the use cases studied and others due to lack of time or because they have been released after the research works were conducted. For instance, regarding the training process of the DNNs, Knowledge Distillation [Gou+20] is the process of transferring knowledge from a large model to a smaller one. It allows deploying much smaller models without applying other optimization techniques. Also related to the training process, Curriculum Learning [Ben+09] is a way of training a model where more difficult aspects of a problem are gradually introduced in such a way that the model is always optimally challenged.

Apart from optimization techniques, there are also new model architectures that are gaining prominence recently and that have not been studied in this thesis, such as Transformers [Vas+17]. Novel works have presented techniques that use Transformers to overcome the limitations presented by inductive convolutional biases in an efficient way. These works have already shown promising results in multiple Computer Vision benchmarks in fields such as Object Detection [Car+20], Video Classification [Wan+17], Image Classification [Par+18] and Image Generation [Dos+20]. Some of these architectures are able to match or outperform SOTA results even when getting rid of convolutional layers and relying solely on self-attention.

Finally, future research should not focus only on using existing techniques and architectures, but it should try also to contribute to the state-of-the-art researching new optimization techniques, new algorithms, and even new network architectures.

4 Publications

4.1 Efficient Large-Scale Face Clustering Using an Online Mixture of Gaussians

- **Authors:** David Montero and Naiara Aginako and Basilio Sierra and Marcos Nieto
- **Journal:** Engineering Applications of Artificial Intelligence
- **Volume:** 114
- **Pages:** 105079
- **Year:** 2022
- **Publisher:** Elsevier

Engineering Applications of Artificial Intelligence
journal homepage: www.elsevier.com

Efficient Large-Scale Face Clustering Using an Online Mixture of Gaussians

David Montero[a,b,**], Naiara Aginako[b], Basilio Sierra[b], Marcos Nieto[a]

[a] *Vicomtech, Mikeletegi 57, 20009 Donostia-San Sebastian, Spain*
[b] *University of the Basque Country, Spain*

ABSTRACT

In recent years, the number of applications demanding real-time face clustering algorithms has increased, especially for security and surveillance purposes. However, state-of-the-art face clustering methods are offline, they need to repeat the whole clustering process every time new data arrives, and thus, they are not suitable for real-time applications. On the other hand, online clustering methods are highly dependent on the order and the size of the data, and they are less accurate than offline methods. To overcome these limitations, we present an online gaussian mixture-based clustering method (OGMC). The key idea of this method is the proposal that an identity can be represented by more than just one distribution or cluster. Using feature vectors extracted from the incoming faces, OGMC generates clusters that may be connected to others depending on their proximity and their robustness, and updates their connections every time their parameters are updated. With this approach, we reduce the dependency of the clustering process on the order and the size of the data and we are able to deal with complex data distributions. Experimental results show that OGMC outperforms state-of-the-art clustering methods on large-scale face clustering benchmarks not only in accuracy, but also in efficiency and scalability.

1. Introduction

Online algorithms are present in a wide variety of applications that affect us in our daily lives, and are essential for their correct and efficient operation (Li and Yu, 2021; Li et al., 2021; Abdalzaher et al., 2021). In recent years, their application has also expanded to the field of facial recognition and clustering. For instance, it is required in real-time video-surveillance applications, where there is a need to maintain an updated database of subjects within the monitored area, either to control their locations (Mahdi et al., 2016) or to re-identify a subject if necessary (Apoorva. et al., 2019; Yimyam et al., 2018). Another usage of face clustering in a real-time application is people flow monitoring in large infrastructures. This type of application aims to generate Performance Indicators, such as ”waiting times”, ”process throughput”, ”person show-up profile”, ”queue length overrun” and ”area occupancy” (Mayer et al., 2015; Chien et al., 2019), which need to be computed for increasingly large number of cameras, and thus demanding a higher level of computation scalability.

An important issue that arises about these types of applications is data privacy (Erkin et al., 2009). Personal information that could be used to identify the subjects should not be stored (i.e., images of their faces). Recent clustering algorithms rely on Deep Neural Networks (DNN) that can infer feature vectors (f-vectors) from the targeted face images, which correspond to abstract representations of the appearances of the people used for training. Although in the last few years some promising methods for face rendering from f-vectors have been released (Lombardi et al., 2018; Duong et al., 2020), they require knowing the feature extraction network in order to generate acceptable results (Duong et al., 2020) or to train the decoding network (Lombardi et al., 2018). Therefore, the individual identities can be protected by hiding and securing the embedding network. Obviously, this approach has limitations, such as the possibility of matching different people that look too similar for the trained model. Therefore, the design and training of the DNN should cover as well as possible the subtle facial appearance dissimilarities. State-of-the-art face recognition models (Deng et al., 2019; Schroff et al., 2015) aim at accomplishing

**Corresponding author: Tel.: +3-467-171-7475;
e-mail: dmontero@vicomtech.org (David Montero)

that goal.

Furthermore, these real-time applications may work in large-scale unconstrained environments, where no information about the distribution of face representations or the number of identities is available. The faces may come from multiple cameras placed in different locations and positions, so they may have different orientations, lighting conditions, partial occlusions, etc. This will lead to complex data distributions. Most traditional and state-of-the-art clustering methods are offline (Wang et al., 2019; Otto et al., 2018; Lloyd, 1982; Jain and Dubes, 1988). These offline approaches are not suitable for real-time large-scale scenarios, as they need to repeat the whole clustering process every time a new sample arrives. On the other hand, the availaible online face clustering methods either rely on spatio-temporal constraints (Kulshreshtha and Guha, 2018) or assuming simple data distributions (Tapaswi et al., 2019), which makes them highly dependent on the order of the data and unable to deal with complex data distributions.

To address these limitations, we propose an online gaussian mixture-based clustering method (OGMC), where we assume that each identity may be represented by more than one distribution or cluster. Thus, OGMC generates clusters that may be connected to others depending on their proximity and robustness. Each group of connected clusters represents an identity, allowing us to deal with complex distributions. Furthermore, every time a cluster is updated, its connections are also updated, reducing the dependency of the process on the order of the data. The high-level idea of the method is exposed in Figure 2. Experimental results show that our approach outperforms state-of-the-art clustering methods on large-scale face clustering benchmarks not only in accuracy, but also in efficiency and scalability, and that it is suitable for large-scale real-time applications.

The rest of the paper is organized as follows. First, we present a review of the related work in section 2. Section 3 describes the proposed clustering method and all the details about its implementation. In section 4 we present the results of a series of experiments conducted to show the capabilities of OGMC and to compare it with the state-of-the-art alternatives. Furthermore, this section also includes an ablation study to discuss some important design choices. Finally, the conclusions and future work directions are given in section 5.

2. Related Work

2.1. Unconstrained Face Clustering

Face clustering in unconstrained environments has become a well-studied topic over the last few years. The addressed scenarios are increasingly complex and encompass a greater number of identities. The huge number of faces and the intra-class appearance changes that might happen due to environmental variations (e.g., pose, illumination, expression, ornaments, occlusions, resolution, image noise) lead to complex distributions of face representations. Traditional clustering algorithms, such as K-Means (Lloyd, 1982) or spectral clustering (Shi and Malik, 2000), suffer from this complexity and they are not able to achieve acceptable performance in these scenarios, as they make assumptions on data distribution. For instance, K-Means tends to generate similar-sized clusters.

The new trends combine face recognition models based on DNN to extract facial f-vectors with sophisticated clustering algorithms that can group them in distinguishable identities, despite the intra-class appearance variability. Shi et al. (2018) proposed the Conditional Pairwise Clustering (ConPaC) algorithm, which is based on the direct estimation of an adjacency matrix using pairwise similarities between f-vectors. In Wang et al. (2019) a linkage-based face clustering algorithm is presented, where a graph convolution network decides which pairs of nodes should be linked. Lin et al. (2018) adopted an Agglomerative Hierarchical Clustering approach, considering the distance measure in the embedded space and the dissimilarity between two groups of faces. In Otto et al. (2018) an approximate rank-order clustering is presented, which predicts whether a node should be linked to its k Nearest Neighbors (kNN), and transitively merges all linked pairs.

Nevertheless, all these state-of-the-art methodologies employ offline algorithms. They process entirely the gathered data every time a new sample arrives, repeating the clustering process with increasing computational cost, in order to get an optimal result. In addition, most of them suffer from scalability problems in terms of accuracy and processing time. For instance, the complexity of ConPaC can scale up to $O(TN^3)$, where N is the number of f-vectors and T the number of iterations. Wang et al. (2019) and Otto et al. (2018) reduce the complexity of the proposed algorithms using kNN graphs to reduce the number of comparisons, but the computational cost is still too high to consider them for online applications.

2.2. Online Clustering

In recent years, numerous online clustering algorithms have emerged to tackle challenging situations. The main advantage of this type of algorithms is that they are able to process a new sample without repeating the whole clustering process. Therefore, they are the best choice when dealing with large-scale real-time scenarios, but with the added challenge of controlling and defining the learning rate (i.e., how new data updates the learnt models).

These types of algorithms have been applied in a wide variety of contexts. For instance, they have gained an increasing importance in text clustering. In Comito et al. (2016), the authors proposed an online clustering method for grouping data streams from social networks by their topic, using a similarity measure computed taking into account both the cluster age and the employed terms. Yin and Wang (2016) presented an alternative text clustering method, assuming an unknown number of clusters, but below a maximum. Online clustering algorithms have also been employed for unsupervised representation learning (Zhan et al., 2020), where the cluster centroids evolve dynamically, keeping the classifier stably updated. Another example of online clustering application is MalFamAware (Pitolli et al., 2020), an algorithm created to group new malwares into families to discern if they are novel or just a variant of a known sample. Even traditional offline methods, such as K-Means, have their own online implementation (Liberty et al., 2016).

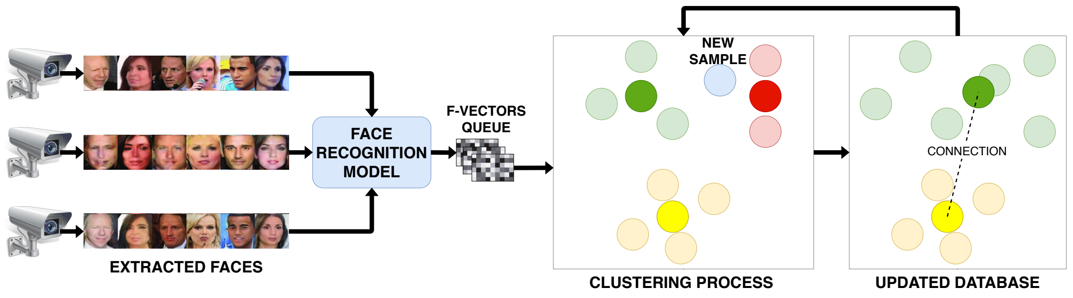


Fig. 1. Example of an online clustering system. A continuous stream of face images, extracted from a set of video-surveillance cameras, are processed by a face recognition model and the extracted f-vectors are enqueued. The online clustering process updates the database with every new sample without repeating the whole process.

Some novel clustering algorithms try to combine both online and offline approaches. Wang and Imura (2019) presented a gaussian process-based incremental neural network, where the new samples are treated as nodes, which may be connected to others. When a new sample is clustered, only the connections of its neighbours are re-evaluated. Meanwhile, in Hyde et al. (2016), samples are grouped in spherical static micro-clusters, connected together to create dynamically shaped macro-clusters. These approaches aim to achieve the same accuracy as offline methods but with an important decrease in the computation time.

In face clustering there are also some online approaches. In Kulshreshtha and Guha (2018), the authors propose an online algorithm for clustering faces in long videos. They process the data sequentially in short segments of variable length and create clusters using face representations and several spatio-temporal constraints. Nevertheless, its reliance on these constraints makes their method unsuitable for unconstrained environments and highly susceptible to the order of the incoming data. Tapaswi et al. (2019) presented another online face clustering method for long videos. The algorithm creates spherical clusters with a shared radius which may vary dynamically. They assume that each identity is represented by an identical distribution. Thus, this method is not able to deal with complex data distributions and, therefore, is not a valid solution for unconstrained environments.

Our proposed method aims to cover the need for an online face clustering algorithm capable of working in large-scale unconstrained environments in real time, achieving state-of-the-art results.

3. Proposed Method

3.1. Problem Definition

We consider the problem of online clustering: given a continuous stream of unknown faces, create a database grouping the incoming faces by their identity. The database must be updated every time a new face arrives, so that information on existing identities is available in real time. An example of an online clustering system is shown in Figure 1. For instance, a real-time video-surveillance application based on face recognition that is operating in a very large infrastructure, such as an airport, where thousands of people pass every day and millions of face images are captured by the cameras. For time and scalability

Table 1. Table of notations for the proposed method.

Notation	Description
F	Clustering process repeated for each sample.
D_i	Updated database for the i-th iteration.
S_i	i-th sample, considering a sample as a normalized N-dimensional f-vector from a face.
$C_{i,j}$	j-th cluster for the i-th iteration.
$I_{i,j}$	Identity of the j-th cluster for the i-th iteration.
ns_r	Minimum number of samples to consider a cluster as robust. Model parameter.
nc_{max}	Maximum number of connections allowed for a robust cluster. Model parameter.
μ	Mean vector of a cluster's distribution.
Σ	Covariance matrix of a cluster's distribution.
σ	Standard deviation of a cluster's distribution.
thr_f	Fusion threshold. Model parameter.
thr_{wc}	Weak connection threshold. Model parameter.
thr_{sc}	Strong connection threshold. Model parameter.
$C*_{i,j}$	Centroid of the j-th cluster for the i-th iteration.
$SC_{i,j}$	Sum of the feature vectors of all the samples of the j-th cluster for the i-th iteration.
$ns_{i,j}$	Number of samples in the j-th cluster for the i-th iteration.
$sIdx_{i,j}$	List of indices of the samples in the j-th cluster for the i-th iteration.
$cIdx_{i,j}$	List of indices of the clusters connected to the j-th cluster for the i-th iteration.
$cIdx_{i,j}$	List of distances to the connected clusters for the j-th cluster for the i-th iteration.
$dist(X, Y)$	Euclidean distance between X and Y.
$dist_M(X, Y)$	Mahalanobis distance between X and Y.
$cosSim$	Cosine similarity between X and Y.

considerations, it is not viable to regenerate the whole database in every iteration, so the algorithm must cluster the new sample using the information from the existing database and update it. Therefore, the problem can be modeled as follows:

$$D_i = F(S_i, D_{i-1})$$
$$D_i = C_i, I_i \tag{1}$$

where D_i is the updated database for the i-th iteration, F is the clustering process repeated in each iteration, S_i is the i-th sample, considering a sample as a normalized N-dimensional f-vector extracted from an incoming face, and D_{i-1} is the resulting database from the previous iteration. The database D is represented by the group of computed clusters C and by the identities I to which they belong. We aim at modeling F maximizing the accuracy and the scalability and minimizing the iteration time. In order to facilitate the readability and the comprehension of the method, we provide a table with the abbreviations and notations for all the parameters and variables, following the design presented in Moustafa et al. (2021a) (see Table 1).

3.2. Expectation-Maximization Approach

The problem modeled by the equation 1 is similar to the one tackled by the Expectation-Maximization (EM) algorithm Moon (1996). The EM algorithm is a well-known iterative approach to perform maximum likelihood estimation in the presence of latent variables. It has been widely used in clustering applications (Zhang and Baek, 2019; Aiadi et al., 2019; Uykan, 2021; Garriga et al., 2015). The problem formulation is the following: given a statistical model generated by a set of observed data X, a set of missing values Z, and a vector of unknown parameters Θ, along with a likelihood function:

$$L(\Theta; X, Z) = p(X, Z|\Theta) \tag{2}$$

the maximum likelihood estimate (MLE) of the unknown parameters is determined by maximizing the marginal likelihood of the observed data:

$$L(\Theta; X) = p(X|\Theta) \tag{3}$$

The EM algorithm aims to find the MLE of the marginal likelihood by iteratively applying two steps:

- Expectation Step: estimates the values of the missing data Z, using the observed data X and the current estimation of the parameters Θ_i.

$$Z = F(\Theta_i, X) \tag{4}$$

- Maximization Step: update the parameters Θ_{i+1} using the observed data X and the new estimated data Z.

$$\Theta_{i+1} = F(X, Z) \tag{5}$$

In our context, we apply the EM algorithm assuming that, at each iteration, the observed data is the group of processed samples S, the parameters are the features of the computed clusters C, and the missing value we want to estimate is the identity of the cluster to which the new sample belongs I.

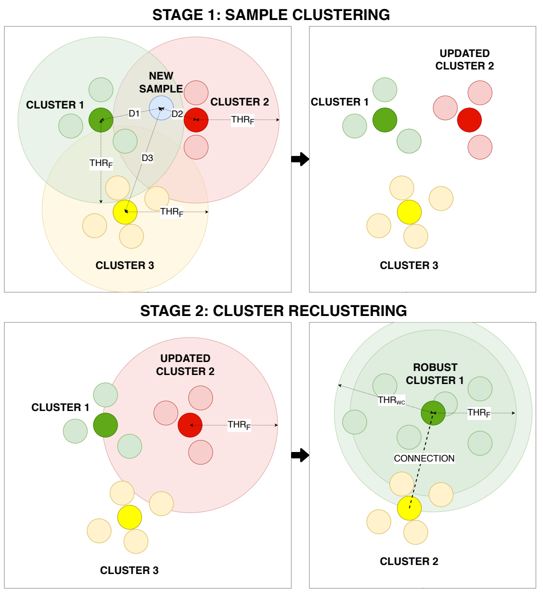


Fig. 2. Example of the operation of the proposed online clustering algorithm when a new sample arrives.

Thus, in the estimation step, using the parameters of the clusters computed with the already processed samples, the algorithm decides whether the new sample should be merged with an existing cluster or a new one should be created. Then, in the maximization step, the parameters of the involved cluster are updated.

Nevertheless, our method follows a variant of this two-steps approach. Every time a cluster is updated, a new EM algorithm is launched, called cluster reclustering, where we check if the updated cluster can be fused with others. This is an iterative process, so a cluster may be fused multiple times. Thus, our algorithm is divided into two stages:

- Sample clustering stage: the new sample is clustered, updating an existing cluster or creating a new one.
- Cluster reclustering stage: iteratively tries to fuse the updated cluster with the rest of the clusters in the database.

The two-stages process is illustrated in Figure 2, and described in subsequent sections.

3.3. Cluster Connections

As we want our algorithm to work in unconstrained environments, it must be able to deal with complex data distributions. Therefore, we must consider the possibility that an identity is represented by more than one cluster. For instance, facial attributes of a person may change for different reasons (i.e., glasses, beard, hair, perspective, lighting conditions, ...), and trying to group all faces in just one cluster may lead to errors due to overly permissive thresholds and poor quality centroids.

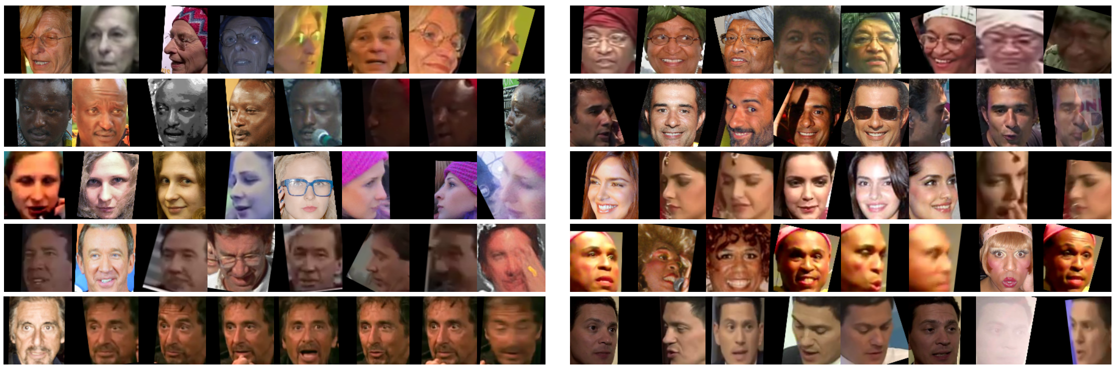


Fig. 3. Examples of complex identities extracted from the IJB-C dataset. Different variations in lighting conditions, occlusions, perspective and facial attributes such as beard, glasses, hat or hair can be observed. Trying to group all faces in just one cluster may lead to errors due to overly permissive thresholds and poor quality centroids.

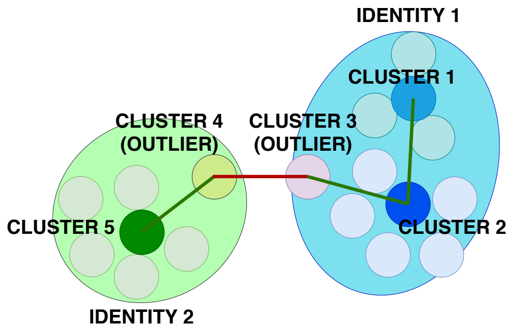


Fig. 4. Example of an erroneous connection caused by outliers. Clusters 1, 2 and 5 are robust while clusters 3 and 4 are non-robust. These problems can be avoided by applying the defined connection rules.

Several examples of complex identities extracted from the IJB-C dataset (Maze et al., 2018) are exposed in Figure 3.

For this reason, we introduce the concept of cluster connection. A connection between two clusters implies that they belong to the same identity, but they represent different distributions in the feature-space. In other words, there is a high similarity between both clusters, but this similarity is not high enough to fuse them (see Figure 2). This way, the algorithm creates trees of connected clusters to deal with complex data distributions.

Furthermore, with these connections we highly reduce the dependency of our algorithm to the order of the incoming samples, as the connections of a cluster are checked and updated every time it is fused with a new sample or with another cluster.

Nevertheless, allowing multiple connections without control may lead to erroneous connections between outliers and clusters belonging to different identities (see Figure 4). To overcome this problem, we create the concept of robust clusters and several rules to control the connections. A cluster is considered robust if it is composed of, at least, ns_r samples, so that we can ensure it is not a group of outliers. The connection rules are the following:

- A non-robust cluster shall have at most one connection, and it can only be connected to a robust cluster. With this rule, we avoid erroneous connections caused by outliers and redundant connections of non-robust members of an identity, solving problems such as that exposed in Figure 4.
- Two robust clusters can not be fused together. We consider a robust cluster as a valid distribution which represents a subset of an identity samples. Therefore, joining two robust clusters would result in a poorer representation of the identity and a loss of information.
- A robust cluster may have a maximum number of connections nc_{max}. This limitation is adopted to reduce the computation time, especially when checking if the connections of a cluster are still valid.
- Every time a cluster is fused or it is connected to new clusters, a process to check connections is triggered. This function checks if the connections of the updated cluster are still valid and that the maximum number of connections is not exceeded. Otherwise, the weakest connections are removed.

3.4. Cluster Representation

The next step in the creation of the clustering algorithm is the selection of the group of parameters that represents each cluster. These parameters must contain enough useful information about the cluster they represent in order to obtain accurate results. Furthermore, this information should be brief and concise, as the algorithm needs to be fast and scalable.

Considering these factors, we decided to model the clusters using multivariate normal distributions, as it only depends on two parameters: the mean vector μ and the covariance matrix Σ. This is a design choice also motivated because the EM algorithm works well with these kind of distributions (Vila and

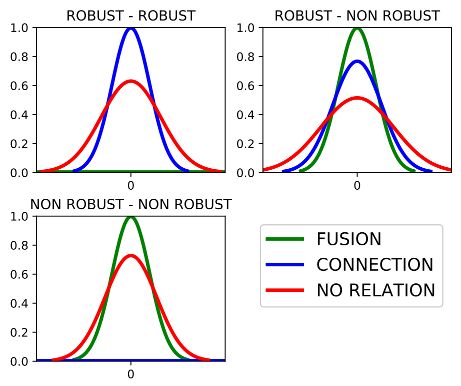


Fig. 5. Normal distributions representing the probability that a cluster can be fused to, connected to or independent of another. The gaussians are centered in 0 (the minimum possible distance between the centroids). The variance depends on whether the compared clusters are robust or not and on the connection rules defined in Section 3.3.

Schniter, 2013; Tian et al., 2011; Guo et al., 2012). Therefore, the density function that models the probability that the N-dimensional sample S_i belongs to cluster j is:

$$P(S_i|C_{i-1,j}) \propto exp\left(-\frac{1}{2}\left(S_i - \mu_{i-i,j}{}^T\right)\Sigma^{-1}_{i-1,j}\left(S_i - \mu_{i-1,j}\right)\right) \quad (6)$$

where the Mahalanobis distance (De Maesschalck et al., 2000) between S_i and $\mu_{i-1,j}$ can be directly used to evaluate which is the closest centroid for a certain sample. Indeed, if we assume all dimensions are independent and have the same variance, we can operate on Mahalanobis distances to reduce the computational load of the algorithm, defined as follows:

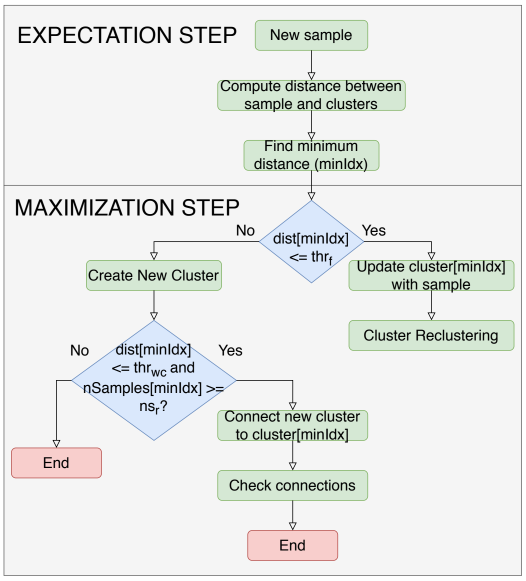
$$dist_M(S_i, \mu_{i-1,j}, \sigma_{i-1,j}) = \frac{1}{\sigma_{i-1,j}} dist(S_i, \mu_{i-1,j}) \quad (7)$$

where $dist(S, \mu) = ||S - \mu||_2$.

The three possible cases when comparing two clusters (fusion, connection or no relation) are then modeled as a mixture of gaussians of these normal distributions, and effectively computed using the Mahalanobis distance. These distributions have all zero mean, which is equal to the minimum possible distance between two centroids. Their deviations σ take fixed values depending on whether the compared clusters are robust or not and on the previously defined connection rules (see Figure 5).

Therefore, we define three euclidean distance thresholds to cover all the possible cases generated by the different σ:

- Fusion threshold thr_f: to decide whether two clusters should be fused together.
- Weak connection threshold thr_{wc}: to connect a robust cluster with a non-robust one.
- Strong connection threshold thr_{sc}: to connect two robust clusters.



EXPECTATION STEP
New sample
Compute distance between sample and clusters
Find minimum distance (minIdx)
MAXIMIZATION STEP
No
dist[minIdx] <= thr_f
Yes
Create New Cluster
Update cluster[minIdx] with sample
Cluster Reclustering
No
dist[minIdx] <= thr_wc and nSamples[minIdx] >= ns_r?
Yes
Connect new cluster to cluster[minIdx]
End
Check connections
End

Fig. 6. Diagram describing the sample clustering stage of the proposed clustering algorithm.

The mean of a cluster $C_{i,j}$ is represented by its centroid $C^*_{i,j}$, computed by normalizing the sum of the features of all the samples contained in the cluster:

$$\begin{aligned} SC_{i,j} &= \sum S \in cluster_{i,j} \\ C^*_{i,j} &= \frac{SC_{i,j}}{||SC_{i,j}||_2} \end{aligned} \quad (8)$$

Thus, the following information is stored for each cluster:

- N-dimensional centroid (C^*)
- Sum of the belonging samples features (SC)
- Number of belonging samples (ns)
- Index of the belonging samples ($sIdx$)
- Index of the connected clusters ($cIdx$)
- Distance to the connected clusters ($cDist$)

3.5. *Method Implementation*

As described in section 3.2, our clustering method is divided into two stages. The first one, the sample clustering stage, is illustrated in Figure 6. In this stage, the new incoming sample is processed. The first step is to compute the distance between the normalized sample vector and the normalized centroids of the existing clusters in the database. Different distances may be considered, but we selected the euclidean distance ($dist$). We made this decision because the employed face recognition model was trained using the cosine similarity (Deng

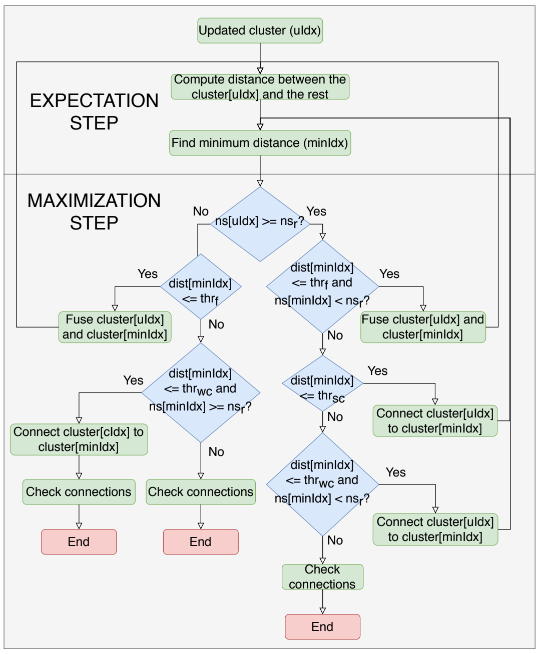

Updated cluster (uIdx)
EXPECTATION STEP
Compute distance between the cluster[uIdx] and the rest
Find minimum distance (minIdx)
MAXIMIZATION STEP
ns[uIdx] >= ns_r?
No
Yes
dist[minIdx] <= thr_f
Yes
No
Fuse cluster[uIdx] and cluster[minIdx]
dist[minIdx] <= thr_f and ns[minIdx] < ns_r?
Yes
No
Fuse cluster[uIdx] and cluster[minIdx]
dist[minIdx] <= thr_wc and ns[minIdx] >= ns_r?
Yes
No
Connect cluster[cIdx] to cluster[minIdx]
Check connections
End
Check connections
End
dist[minIdx] <= thr_sc
Yes
No
Connect cluster[uIdx] to cluster[minIdx]
dist[minIdx] <= thr_wc and ns[minIdx] < ns_r?
Yes
No
Connect cluster[uIdx] to cluster[minIdx]
Check connections
End

Fig. 7. Diagram describing the cluster reclustering stage of the proposed clustering algorithm.

et al., 2019) and, for normalized vectors, it is inversely proportional to the euclidean distance:

$$dist(V1, V2) = \sqrt{2(1 - cosSim(V1, V2))} \tag{9}$$

The distances are computed in parallel taking advantage of the capabilities of a GPU architecture. This way, we reduce the impact of the number of clusters in the processing time. For this reason, the normalized centroids are stored directly in the GPU memory.

Once the distances have been computed, they are copied to the RAM memory and the minimum distance is selected using the CPU. If this distance is less than thr_f, the sample is fused with the selected cluster. Otherwise, it is used to create a new cluster. If a new cluster is created, the algorithm checks if it can be connected to the minimum distance cluster. This connection happens if the selected cluster is robust and if the distance is not higher than thr_{wc}. If there is a connection, the algorithm checks if the number of connections of the robust cluster has exceeded the maximum permitted (nc_r), and, if necessary, erases the weakest connections.

If the new sample S_i is used to update an existing cluster, the cluster reclustering stage is launched (see Figure 7). This stage is composed of an iterative EM algorithm, where an attempt is made to fuse the updated cluster or connect it with the rest of the clusters in the database, using the parameters thr_f, thr_{wc}, thr_{sc} and ns_r, and the connection rules defined in section 3.3.

The selection of the minimum distance is computed in the same way as in the previous stage. If the updated cluster is fused with another one, the distances are computed again for the new updated cluster and the process is repeated. If the updated cluster is connected to another and the updated cluster is robust, the algorithm searches for the next minimum distance and repeats the maximization step.

Before the cluster reclustering stage ends, the algorithm checks if the connections of the updated cluster are still valid, as the centroid of the cluster may have changed due to a fusion. Finally, it checks that the number of connection of the updated cluster and of the connected clusters do not exceed the maximum number of connections allowed.

The results of the clustering process are extracted after each iteration using a simple recursive function and the variables $sIdx$ and $cIdx$ to merge the samples of all connected clusters.

4. Experiments

A series of experiments have been conducted to demonstrate the potential of the proposed clustering algorithm OGMC. The experiments are divided into two groups. The first group of experiments aims to measure the performance of OGMC in accuracy and processing time, comparing it with other traditional and state-of-the-art offline clustering methods.

The second group focuses on testing its scalability, measuring the drop in accuracy and the increase in processing time as the number of data samples grows.

Furthermore, an ablation study is presented in order to validate the design decisions, compare the contribution of the different parts of the algorithm and measure the degree of dependency of the model parameters on the face recognition network and the train and test datasets.

Finally, to demonstrate the effectiveness of OGMC beyond face recognition, an additional experiment is conducted with DeepFashion (Liu et al., 2016), a well-known dataset used for clothes retrieval.

The server used to carry out the experiments was equipped with an NVIDIA Tesla V100 GPU and an Intel Xeon Gold 6230 CPU. For all the experiments, and for all the tested methods, the server status was idle, without any other process running in parallel.

4.1. Parameter Tuning

As discussed in section 3, the OGMC depends only on 5 parameters: three distance thresholds (thr_f, thr_{wc}, thr_{sc}), the minimum number of samples to classify a cluster as robust (ns_r) and the maximum number of connections allowed for a robust cluster (nc_r). This number of parameters is relatively low taking into account that it is an online method which can operate with a database regardless of its magnitude, and compared to other state-of-the-art clustering methods, such as, for example, GCN (Wang et al., 2019), which requires training a DNN in addition to three parameters. Furthermore, their values are bounded within certain limits and they are easy to adjust, as is explained below.

The distance thresholds are floating-point numbers bounded between 0.0 and 2.0, as they represent the euclidean distance between two normalized vectors. These are, by far, the most sensitive parameters of the algorithm, as small modifications of their values have significant impact on the output.

The minimum number of samples to classify a cluster as robust (ns_r) is an integer equal to or greater than 1. On the one hand, the lower its value, the higher the number of robust clusters, which may be undesirable because of errors in connections between outliers of different identities and an increase in the processing time due to circular connections. On the other hand, if ns_r is too high, the number of robust clusters may not be enough to connect all the non-robust clusters together. We have empirically found that a range between 3 and 6 puts the algorithm into equilibrium, and thus suggests that the optimal value is 4, so the user of the algorithm does not really need to change it.

The maximum number of connections allowed for a robust cluster (nc_r) is also an integer equal to or greater than 1. It needs to be adjusted along with ns_r, because if ns_r decreases, so does the number of robust clusters, and thus the number of connections required per robust cluster increases. If the distance thresholds are set correctly, the accuracy should not be affected by increasing the number of allowed connections. However, this parameter may have a small impact on the processing time by limiting the number of circular connections between clusters of the same identity. We have empirically determined that the appropriate range of values for this parameter is between 5 and 25 and that it can be tuned with a sensitivity of 5.

These parameters, and especially the distance thresholds, depend mainly on the face recognition model, so they only need to be readjusted if the model is replaced. Although they may also have a small dependency on the employed dataset, it has a limited impact on the results, as we report in the subsequent ablation study.

The steps followed to fine tune the parameters are:

- Iterative grid-search for tuning the three distance thresholds. The grid size starts with a size of 0.1 and is halved on each iteration until it reaches a resolution of 0.0125 (4 iterations). The total number of tests is limited according to the restrictions: thr_{wc} must be smaller than thr_f and that thr_f must be smaller than thr_{sc}. In this step, ns_r and nc_r take fixed values of 4 and 10 respectively.
- Single grid-search for tuning ns_r and nc_r. For ns_r, use a grid size of 1 between 3 and 6 and for nc_r a grid size of 5 between 5 and 25 (20 cells in total). In this step, the distance thresholds are fixed and take the values computed in the previous step.

Exploring the parameter space with the suggested approach and testing the response of the algorithm in accuracy and processing time against a training dataset, the user can easily configure the algorithm according to the requirements of the application. Furthermore, we parallelize every grid search in order to reduce the tuning time.

Finally, the dataset selected for tuning the parameters is reported in the description of each experiment.

4.2. *Face Clustering Performance*

For the first experiment, we use the IJB-B dataset (Whitelam et al., 2017), a well-known dataset of unconstrained in-the-wild face images. This dataset includes a clustering protocol consisting of seven subtasks that vary in the number of identities and the number of faces. We select the last subtask, as it is the most challenging one, with the highest number of identities (1,845) and faces (68,195).

For a fair comparison with other methods, and to demonstrate the OGMC algorithm is independent of the recognition model, we use the same vectors as in Wang et al. (2019) for the experiment. These vectors have 512 dimensions and have been generated using a face recognition model based on ArcFace (Deng et al., 2019). The training dataset consists of a random subset from the CASIA dataset (Yi et al., 2014), with 5,000 identities and 200,000 samples. Thus, we tuned the model parameters using this training dataset: $thr_f = 1.01$, $thr_{wc} = 1.12$, $thr_{sc} = 0.99$, $ns_r = 5$, $nc_r = 5$.

The performance is measured following the recommendations in Amigó et al. (2009), selecting the following metrics:

- BCubed F-Measure F: represents the clustering system effectiveness, taking the bcubed precision P and recall R into account, which are computed as described in Amigó et al. (2009).
$$F = 2\frac{P * R}{P + R} \tag{10}$$
- Normalized Mutual Information (NMI): this measure represents the homogeneity of the clusters. Using the ground truth clusters (G) and the predicted clusters (C) it can be computed with the following equation:
$$NMI(G, C) = \frac{I(G, C)}{\sqrt{H(G)H(C)}} \tag{11}$$
where H represents the entropy and I is the mutual information.
- Total processing time: since we are comparing our online algorithm with other offline methods, we process all the samples sequentially to simulate an offline behaviour and we compute the time for the whole process.

The results of the experiment are presented in Table 2. It can be observed that the proposed method outperforms the others in terms of F-Measure and processing time, while achieving competitive results in the cluster homogeneity measure. Compared to the second best method (GCN-A), OGMC achieves a better F-Measure while reducing the processing time by more than 6 times using the same hardware.

In the second experiment, we test the performance of our method using a different face recognition model and a different face dataset. The employed face recognition model is trained using ArcFace loss (Deng et al., 2019), with ResNet100 (Han et al., 2017; He et al., 2016) as the embedding network, an input resolution of 112×112 and an output embedding dimension of 512. We select MS1MV2 (Deng et al., 2019) as the training dataset, which is a refinement of MS-Celeb-1M (Guo et al., 2016).

The dataset selected for this experiment is IJB-C (Maze et al., 2018), another well-known dataset of unconstrained in-the-wild face images, with a higher number of identities and faces. This

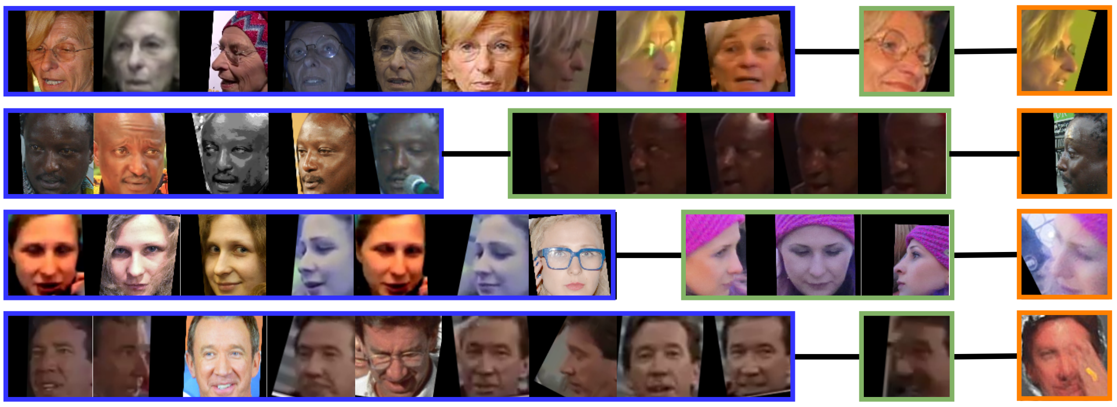


Fig. 8. Example clusters and connections generated by the proposed method in the IJB-C experiment. Each row represents an identity composed of several clusters connected together.

Table 2. Comparison with baseline methods in terms of BCubed F-Measure, Normalized Mutual Information (NMI) and processing time using the IJB-B 1845 subtask. Superscript* denotes results reported from the original papers, otherwise all methods use the f-vectors from Wang et al. (2019). Superscript^T denotes times reported from Lin et al. (2018). The methods considered for the comparison are K-Means (Lloyd, 1982), Spectral (Shi and Malik, 2000), AHC (Jain and Dubes, 1988), AP (Frey and Dueck, 2007), DBSCAN (Ester et al., 1996), ARO (Otto et al., 2018), PAHC (Lin et al., 2018), ConPaC (Shi et al., 2018), DDC (Lin et al., 2018) and GCN-A (Wang et al., 2019).

Method	F-Measure	NMI	Run-Time
K-Means^T	0.600	0.868	00:01:00
Spectral^T	0.516	0.785	-
AHC^T	0.793	0.923	00:01:32
AP^T	0.477	0.869	08:42:50
DBSCAN^T	0.695	0.814	00:49:31
ARO^T	0.755	0.913	00:01:13
PAHC*^{T}	0.610	0.890	00:03:56
ConPaC*^{T}	0.634	-	02:53:58
DDC^T	0.800	0.929	00:05:32
GCN-A	0.814	**0.938**	00:06:03
OGMC (ours)	**0.822**	0.921	**00:00:55**

dataset also includes a clustering protocol with 8 subtasks. Again, we select the most challenging protocol (IJB-C-3531), with 3,531 identities and 140,623 faces. We use RetinaFace (Deng et al., 2020) for the face and facial landmarks detection. In order to obtain better quality feature vectors, we filter faces with less than 45 pixels per side and we normalize the face patches applying an affine transformation using reference facial landmarks, as recommendend in Wang et al. (2018). After filtering, 120,661 vectors belonging to 3,529 identities are extracted.

We compare our algorithm with the two best state-of-the-art methods: GCN (Wang et al., 2019), which achieved the highest accuracy in the first experiment (without considering ours),

Table 3. Comparison with baseline methods in terms of BCubed F-Measure (F-Meas), Dunn's index (Dunn), Caliński-Harabasz index (C-H), Silhouette index (Silh), and processing time using IJB-C feature vectors. All methods use the same vectors and hardware.

Method	F-Meas	Dunn	C-H	Silh	Run-Time
ARO	0.768	0.036	17.707	0.075	00:09:39
GCN	0.906	0.040	16.334	0.161	00:10:32
OGMC	**0.948**	**0.116**	**34.932**	**0.171**	**00:01:32**

and ARO (Otto et al., 2018), which achieved the best trade-off between accuracy and speed (without considering ours). We adjust the parameters of both methods to achieve the best performance. Furthermore, for GCN, we retrain the network using a random subset of VGG2 dataset (Cao et al., 2018), containing over 300k images and 8500 identities, during 4 epochs (following the recommendations in Wang et al. (2019)). Finally, we readjust the parameters of our algorithm, as we are using a different face recognition model, using the same subset of VGG2 employed for retraining GCN: thr_f = 1.07, thr_{wc} = 1.15, thr_{sc} = 1.05, ns_r = 5, nc_r = 10.

In order to perform a more exhaustive comparison between the methods, for this experiment we also compute three additional metrics that have been specifically developed for the evaluation of clustering algorithms: Dunn's index, Caliński-Harabasz index, and Silhouette index, which are described in Moustafa et al. (2021a). For the three indices, higher values mean better results. The results of this experiment, presented in Table 3, show that our method outperform the others in all the metrics and, specially, in the processing time. OGMC runs 7 times faster than GCN and more than 6 times faster than ARO. These ratios of processing time show that our method is also more suitable to scale-up compared to the others.

We also present several example clusters generated by the proposed method during the experiment. In Figure 8, each row represents an identity composed of different clusters connected to each other. These identities contain variations in lighting conditions, partial occlusions, perspective and facial attributes, so they are represented by complex data distributions. With our proposed method, we are able to accurately approximate these distributions using a variable number of connected clusters. Each robust cluster generates a centroid (representative f-vector) which represents the identity under certain condition range. For instance, in the second row of Figure 8, the second cluster centroid represents the identity under dim lighting conditions. If we tried to group all faces of this identity in just one cluster, it could lead to errors due to overly permissive thresholds and worse quality centroids. Another benefit of OGMC that can be observed in this figure is that the identity outliers are contained in non-robust clusters connected to the robust ones. This way, the quality of the robust cluster centroids is not compromised by these outliers and the number of matching errors are reduced. A clear example of this behaviour is exposed in the last row of Figure 8, where the second and the third clusters are outliers generated by a combination of unsuitable conditions (lighting, blurring, occlusion...).

In the last experiment of this section we test the accuracy of OGMC under extreme conditions: using a face recognition model which extracts vectors with fewer features and using a database with a much higher number of identities. Again, as we want a fair comparison with other methods, we decide to replicate the experiment proposed in Yang et al. (2020). In this experiment, they selected a subset of the database MS-Celeb-1M (Guo et al., 2016) that contains 5.8M images from 86K identities and randomly split it into 10 parts with an almost equal number of identities. Then, they randomly selected 1 part as labeled data for training and the other 9 parts as unlabeled data. With the unlabeled data, they created 5 tests with an increasing number of vectors and identities. The last test has 5.2M vectors and 77K identities. Furthermore the provided vectors have only 256 features, compared to the 512 previously used.

Thus, as they did with the rest of the methods, we tune the parameters of OGMC using the provided training data: thr_f = 0.85, thr_{wc} = 1.02, thr_{sc} = 0.72, ns_r = 4, nc_r = 25. Then, we evaluate the algorithm using the 5 test subsets with an increasing scale of vectors and identities. The results of the experiment are presented in Table 4. It can be observed that OGMC outperforms the rest of the methods consistently in every test subset.

With this experiment, we also prove that the parameters of OGMC do not need to be readjusted if the scale of the dataset increases, so the user just needs to tune the parameters if the face recognition model is changed, as discussed above. Of course, as for any other method, we must ensure the training data is large and varied enough to extract the necessary information from the model to correctly tune the parameters.

4.3. Clustering Scalability

We also want to prove that our method is scalable and that it is suitable for large-scale real-time applications. For this reason, we conduct the following two experiments. In the first one, we test the scalability of our method adding 1, 2 and 3 millions of distractors to the IJB-C vectors used in the previous experiment. Thus, we can observe how the accuracy and the processing time evolve with the size of the database. We generate the distractors from the VGG2 face dataset (Cao et al., 2018), which contains 3.3 millions faces belonging to more than 9000 identities, with large variations in pose, age, illumination, ethnicity and profession. The same face recognition model of the previous IJB-C experiment is used, so we use the same parameter values: thr_f = 1.04, thr_{wc} = 1.13, thr_{sc} = 1.03, ns_r = 5, nc_r = 5.

Tests are executed 10 times, shuffling all the vectors a random number of times and computing the average F-Measure and processing time for all tests. For the computation of the F-Measure, the distractors are ignored. The results shown in Table 5, demonstrate that OGMC has an extremelly high scalability, with a drop in accuracy of 0.1% when adding 1 million distractors and 0.5% when adding 3 million. Furthermore, OGMC is able to process more than 1.1 million faces in less than 25 minutes and more than 3.1 million faces in 2 hours and 13 minutes. Some examples of the clusters and connections generated in this experiment are shown in Figure 11.

Table 4. Comparison with baseline methods in terms of BCubed F-Measure using subsets of different sizes from the MS-Celeb-1M dataset. All methods use the same 256-dimensional vectors provided by Yang et al. (2020). The methods considered for the comparison are K-means (Lloyd, 1982; Sculley, 2010), HAC (Sibson, 1973), DBSCAN (Ester et al., 1996), ARO (Otto et al., 2018), CDP (Zhan et al., 2018), GCN (Wang et al., 2019), LTC (Yang et al., 2019), GCN-V (Yang et al., 2020) and GCN-(V+E) (Yang et al., 2020).

Method	Test				
	Number of samples				
	584K	1.74M	2.89M	4.05M	5.21M
	Number of identities				
	8.5K	25.7K	42.8K	60.0K	77.1K
	BCubed F-Measure				
K-means	0.812	0.752	0.723	0.706	0.694
HAC	0.705	0.695	0.686	0.677	0.670
DBSCAN	0.672	0.665	0.663	0.449	0.447
ARO	0.170	0.124	0.110	0.105	0.100
CDP	0.787	0.758	0.746	0.736	0.729
GCN	0.844	0.816	0.801	0.793	0.786
LTC	0.855	0.830	0.811	0.798	0.789
GCN-V	0.858	0.826	0.811	0.799	0.791
GCN-(V+E)	0.861	0.828	0.812	0.801	0.793
OGMC	**0.906**	**0.881**	**0.864**	**0.851**	**0.839**

Table 5. Results of the proposed method on IJB-C experiment adding distractors from VGG2 dataset.

Samples Number	F-Measure	Run-Time
IJB-C	0.952	00:01:32
IJB-C + 1 million	0.951	00:24:55
IJB-C + 2 millions	0.948	01:00:40
IJB-C + 3 millions	0.947	02:13:02

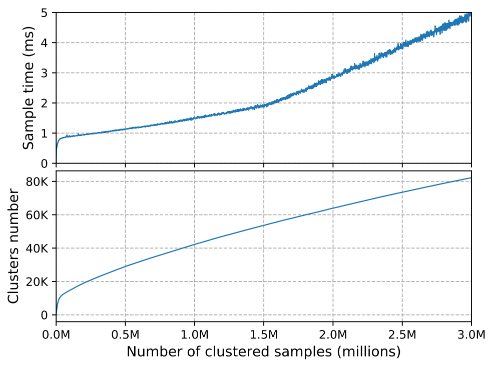


Fig. 9. Evolution of the clustering time per sample with the number of samples clusterized compared to the evolution of the number of clusters in the database.

The last experiment aims to demonstrate that OGMC is suitable for real-time applications. It consists of repeating the previous test using IJB-C vectors with 3 million distractors and measuring how the processing time per sample evolves with the number of processed samples and with the number of clusters in the database (note that the number of clusters is not equal to the number of identities, as we are not taking connections into account).

As in the previous experiment, the test is repeated 10 times, shuffling all the vectors a random number of times and computing the results as the average value of all tests. Figure 9 shows that, after processing 3 million samples and with a database of 80 thousand clusters, the processing time per sample is still 5 milliseconds, which still allows OGMC to process 200 samples per second in real-time. These results demonstrate that the proposed method is suitable for real-time applications, even when dealing with extremely large amounts of data.

4.4. Ablation Study

We present an ablation study to discuss the impact of design choices on the performance of OGMC. First, we demonstrate the contribution of the reclustering stage of the algorithm. For this purpose, we run the IJB-C with VGG2 distractors experiment applying only the first stage of OGMC and compare the results with the ones obtained applying the full method. Table 6 shows the results of the experiment. Comparing the processing times, we can see that removing the reclustering stage significantly reduces the complexity of the algorithm. However, it also leads to an important drop in accuracy (≈4%). Furthermore, removing the second stage also significantly increases the dependency of accuracy on the order of the input data. This is because the connections are not re-evaluated when the clusters are updated.

We also want to analyze the increase in speed added by the GPU parallelization. We reimplement the distance computing module with CPU and repeat the IJB-C experiment with both versions of OGMC to measure how the processing time evolves with the number of clusterized samples in each case. The results of the comparison in Figure 10 show that using the GPU for the distance computing parallelization produces a huge decrease in the computation time per sample and a huge increase in the scalability. However, this simple task only consumes a small percentage of the GPU utilization compared to other methods based on neural networks, such as GCN (Wang et al., 2019) or GCN-V (Yang et al., 2020). Furthermore, the GPU memory needed for the cluster centroids is very limited (2.4GB for 1 million 512-dimensional vectors) and can be allocated dynamically as the database grows, which makes OGMC suitable to run together with other GPU-based algorithms (for example a DNN-based face recognition model).

Finally, we want to demonstrate that the parameters of OGMC do not have a significant dependency on the training dataset, as long as it is large and varied enough to extract the necessary information from the face recognition model. In order to prove this, we repeat the IJB-C experiment with the same face recognition model but, instead of using the VGG2 subset, we tune the parameters using the IJB-C vectors, so that we obtain the best possible results in this test. Thus, we obtain the following values for the parameters: $thr_f = 1.04$, $thr_{wc} = 1.13$, $thr_{sc} = 1.03$, $ns_r = 5$, $nc_r = 5$. With these values OGMC achieves a BCubed F-Measure of 0.952, only a 0.4% of improvement over the original test. Thus, this experiment suggests our hypothesis is correct, the OGMC parameters depend almost

Table 6. Comparison of the first stage of OGMC against the full method.

Dataset	Full method		First Stage	
	F-Meas	Run-Time	F-Meas	Run-Time
IJB-C	0.952	00:01:32	0.914	00:00:49
IJB-C + 1M	0.951	00:24:55	0.912	00:10:12
IJB-C + 2M	0.948	01:00:40	0.907	00:23:11
IJB-C + 3M	0.947	02:13:02	0.901	00:59:18

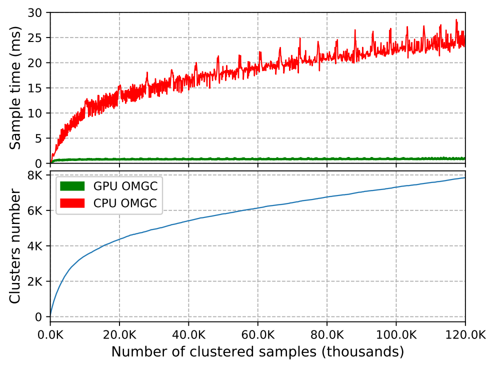


Fig. 10. Comparison of the evolution of the clustering time per sample with the number of samples clusterized for GPU and CPU versions of OGMC.

Table 7. Comparison with baseline methods in terms of BCubed F-Measure with DeepFashion dataset. All methods use the same vectors from Yang et al. (2020). The methods considered for the comparison are K-Means (Lloyd, 1982; Sculley, 2010), HAC (Sibson, 1973), DBSCAN (Ester et al., 1996), MeanShift Yizong Cheng (1995); Comaniciu and Meer (1999), Spectral Ho et al. (2003); Ng et al. (2001), ARO (Otto et al., 2018), CDP (Zhan et al., 2018), GCN (Wang et al., 2019), LTC (Yang et al., 2019), GCN-V (Yang et al., 2020) and GCN-(V+E) (Yang et al., 2020).

Method	F-Measure
K-Means	0.538
HAC	0.488
DBSCAN	0.532
MeanShift	0.567
Spectral	0.464
ARO	0.530
CDP	0.578
GCN	0.589
LTC	0.591
GCN-V	0.573
GCN-(V+E)	0.601
OGMC (ours)	**0.620**

entirely on the face recognition model and they only need to be readjusted if this model is replaced.

4.5. Beyond Face Recogniton

To conclude the experimental section, we evaluate the effectiveness of OGMC for tasks beyond face recognition. For this purpose, we reproduce the experiment presented in Yang et al. (2020) with DeepFashion dataset (Liu et al., 2016), a well-known dataset for clothes retrieval. In this experiment, they mix the training and testing features in the original split, and randomly sample 25,752 images from 3,997 categories for training and the other 26,960 images with 3,984 categories for testing. For a fair comparison, we use the same 256-dimensional vectors provided in their official repository. Thus, we tune the parameters of OGMC using the training set: $thr_f = 0.51$, $thr_{wc} = 0.58$, $thr_{sc} = 0.49$, $ns_r = 4$, $nc_r = 10$. The results of the experiment are presented in Table 7. Among all the tested methods, OGMC achieves the best F-Measure, demonstrating its suitability for tasks beyond face recognition.

5. Conclusions and Future Work

In this work, we address the problem of large-scale online face clustering. We propose a method based on EM algorithm and Mixture of Gaussians. We introduce the concept of cluster connection, where an identity can be represented by multiple clusters. With this approach, we reduce the dependency of the clustering process on the order and the size of the incoming data and we are able to deal with complex data distributions. The conducted experiments and their derived results show that our method outperforms the state-of-the-art clustering methods, regardless of whether they are online or offline, in terms of accuracy, processing time and scalability.

The proposed model was successfully deployed as part of a real-time video-surveillance application based on face recognition. This solution was operating in an airport, where thousands of people were passing every day and millions of face images were captured by the cameras.

Future work will focus on further reducing the processing time of the proposed method by also parallelizing the minimum distance search using the computational capabilities of the GPU. Furthermore, in this work we have proposed a simple method for tuning the parameters of the model, which can be highly optimized or replaced by other more suitable and efficient approaches (Moustafa et al., 2021b,c; Feurer and Hutter, 2019). Although we believe that this task is out of the scope of the current article, we will work on it as future enhancements. Another interesting research line we are considering is to apply additional outlier detection techniques, such as the ones employed in Aiadi et al. (2019), which would make the algorithm more suitable for applications that are more focused on extracting mean KPIs than on the identities themselves (for instance, for counting the number of clients of a supermarket per month). In addition, given the current situation with Covid-19, we will study the impact of the use of surgical masks on the clustering process. We believe that, for an identity, the proposed method would group faces with and without masks in different clusters, but connected to each other, since we have proved that the algorithm works well with partial occlusions. Finally, as we believe in the open source community, we will soon release the full code of OGMC and all the supplementary material used for the experiments, so anyone can replicate them and contribute to improve the method.

References

Abdalzaher, M.S., Soliman, M.S., El-Hady, S.M., Benslimane, A., Elwekeil, M., 2021. A deep learning model for earthquake parameters observation in iot system-based earthquake early warning. IEEE Internet of Things Journal , 1–1doi:10.1109/JIOT.2021.3114420.

Aiadi, O., Kherfi, M.L., Khaldi, B., 2019. Automatic date fruit recognition using outlier detection techniques and gaussian mixture models. ELCVIA Electronic Letters on Computer Vision and Image Analysis 18, 52. doi:10.5565/rev/elcvia.1041.

Amigó, E., Gonzalo, J., Artiles, J., Verdejo, M., 2009. Amigó e, gonzalo j, artiles j et ala comparison of extrinsic clustering evaluation metrics based on formal constraints. inform retriev 12:461-486. Information Retrieval 12, 461–486. doi:10.1007/s10791-008-9066-8.

Apoorva., P., Impana., H.C., Siri., S.L., Varshitha., M.R., Ramesh., B., 2019. Automated criminal identification by face recognition using open computer vision classifiers, in: 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), pp. 775–778.

Cao, Q., Shen, L., Xie, W., Parkhi, O.M., Zisserman, A., 2018. Vggface2: A dataset for recognising faces across pose and age, in: 2018 13th IEEE International Conference on Automatic Face Gesture Recognition (FG 2018), pp. 67–74.

Chien, K.M., Wu, T.C., Luor, T., 2019. Face recognition and smart people-counting system: Cases of asian trade shows. Journal of Internet Technology 20, 435–446. URL: https://jit.ndhu.edu.tw/article/view/2017.

Comaniciu, D., Meer, P., 1999. Mean shift analysis and applications, in: Proceedings of the Seventh IEEE International Conference on Computer Vision, pp. 1197–1203 vol.2. doi:10.1109/ICCV.1999.790416.

Comito, C., Pizzuti, C., Procopio, N., 2016. Online clustering for topic detection in social data streams, in: 2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI), pp. 362–369.

De Maesschalck, R., Jouan-Rimbaud, D., Massart, D., 2000. The mahalanobis distance. Chemometrics and Intelligent Laboratory Systems 50, 1–18. doi:10.1016/S0169-7439(99)00047-7.

Deng, J., Guo, J., Ververas, E., Kotsia, I., Zafeiriou, S., 2020. Retinaface: Single-shot multi-level face localisation in the wild, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

Deng, J., Guo, J., Xue, N., Zafeiriou, S., 2019. Arcface: Additive angular margin loss for deep face recognition, in: 2019 IEEE/CVF Conf. on Computer Vision and Pattern Recognition, pp. 4685–4694. doi:10.1109/CVPR.2019.00482.

Duong, C.N., Truong, T.D., Quach, K.G., Bui, H., Roy, K., Luu, K., 2020. Vec2face: Unveil human faces from their blackbox features in face recognition. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) , 6131–6140.

Erkin, Z., Franz, M., Guajardo, J., Katzenbeisser, S., Lagendijk, I., Toft, T., 2009. Privacy-preserving face recognition, in: Goldberg, I., Atallah, M.J. (Eds.), Privacy Enhancing Technologies, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 235–253.

Ester, M., Kriegel, H.P., Sander, J., Xu, X., 1996. A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, AAAI Press. p. 226–231.

Feurer, M., Hutter, F., 2019. Hyperparameter Optimization. Springer International Publishing, Cham. pp. 3–33. URL: https://doi.org/10.1007/978-3-030-05318-5_1, doi:10.1007/978-3-030-05318-5_1.

Frey, B., Dueck, D., 2007. Clustering by passing messages between data points. Science (New York, N.Y.) 315, 972–6. doi:10.1126/science.1136800.

Garriga, J., Palmer, J., Oltra, A., Bartumeus, F., 2015. Expectation-maximization binary clustering for behavioural annotation. Mov Ecol 11. doi:10.1371/journal.pone.0151984.

Guo, C., Fu, H., Luk, W., 2012. A fully-pipelined expectation-maximization engine for gaussian mixture models, in: 2012 International Conference on Field-Programmable Technology, pp. 182–189.

Guo, Y., Zhang, L., Hu, Y., He, X., Gao, J., 2016. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition, in: ECCV, pp. 87–102. doi:10.1007/978-3-319-46487-9_6.

Han, D., Kim, J., Kim, J., 2017. Deep pyramidal residual networks, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6307–6315.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778. doi:10.1109/CVPR.2016.90.

Ho, J., Ming-Husang Yang, Jongwoo Lim, Kuang-Chih Lee, Kriegman, D., 2003. Clustering appearances of objects under varying illumination conditions, in: 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings., pp. I–I. doi:10.1109/CVPR.2003.1211332.

Hyde, R., Angelov, P., Mackenzie, A., 2016. Fully online clustering of evolving data streams into arbitrarily shaped clusters. Information Sciences 382, 1–41. doi:10.1016/j.ins.2016.12.004.

Jain, A.K., Dubes, R.C., 1988. Algorithms for Clustering Data. Prentice-Hall, Inc., USA.

Kulshreshtha, P., Guha, T., 2018. An online algorithm for constrained face clustering in videos, in: 2018 25th IEEE International Conference on Image Processing (ICIP), pp. 2670–2674.

Li, J., Yu, T., 2021. A new adaptive controller based on distributed deep reinforcement learning for pemfc air supply system. Energy Reports 7, 1267–1279. doi:https://doi.org/10.1016/j.egyr.2021.02.043.

Li, J., Yu, T., Zhang, X., 2021. Emergency fault affected wide-area automatic generation control via large-scale deep reinforcement learning. Engineering Applications of Artificial Intelligence 106, 104500. doi:https://doi.org/10.1016/j.engappai.2021.104500.

Liberty, E., Sriharsha, R., Sviridenko, M., 2016. An algorithm for online k-means clustering, in: ALENEX.

Lin, W., Chen, J., Castillo, C.D., Chellappa, R., 2018. Deep density clustering of unconstrained faces, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8128–8137.

Liu, Z., Luo, P., Qiu, S., Wang, X., Tang, X., 2016. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1096–1104. doi:10.1109/CVPR.2016.124.

Lloyd, S.P., 1982. Least squares quantization in pcm. IEEE Trans. Inf. Theory 28, 129–136.

Lombardi, S., Saragih, J.M., Simon, T., Sheikh, Y., 2018. Deep appearance models for face rendering. ACM Transactions on Graphics (TOG) 37, 1 – 13.

Mahdi, F., Habib, M., Moslehuddin, A., Vasant, P., Mckeever, S., Ahad, M.A.R., 2016. Face recognition-based real-time system for surveillance. Intelligent Decision Technologies 11, 1–14. doi:10.3233/IDT-160279.

Mayer, C.A., Felkel, R., Peterson, K., 2015. Best practice on automated passenger flow measurement solutions, in: Journal of Airport Management, pp. 144–153.

Maze, B., Adams, J., Duncan, J.A., Kalka, N., Miller, T., Otto, C., Jain, A.K., Niggel, W.T., Anderson, J., Cheney, J., Grother, P., 2018. Iarpa janus benchmark - c: Face dataset and protocol, in: 2018 International Conference on Biometrics (ICB), pp. 158–165. doi:10.1109/ICB2018.2018.00033.

Moon, T.K., 1996. The expectation-maximization algorithm. IEEE Signal Processing Magazine 13, 47–60.

Moustafa, S.S.R., Abdalzaher, M.S., Khan, F., Metwaly, M., Elawadi, E.A., Al-Arifi, N.S., 2021a. A quantitative site-specific classification approach based on affinity propagation clustering. IEEE Access 9, 155297–155313. doi:10.1109/ACCESS.2021.3128284.

Moustafa, S.S.R., Abdalzaher, M.S., Yassien, M.H., Wang, T., Elwekeil, M., Hafiez, H.E.A., 2021b. Development of an optimized regression model to predict blast-driven ground vibrations. IEEE Access 9, 31826–31841. doi:10.1109/ACCESS.2021.3059018.

Moustafa, S.S.R., Abdalzaher, M.S., Yassien, M.H., Wang, T., Elwekeil, M., Hafiez, H.E.A., 2021c. Development of an optimized regression model to predict blast-driven ground vibrations. IEEE Access 9, 31826–31841. doi:10.1109/ACCESS.2021.3059018.

Ng, A.Y., Jordan, M.I., Weiss, Y., 2001. On spectral clustering: Analysis and an algorithm, in: Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic, MIT Press, Cambridge, MA, USA. p. 849–856.

Otto, C., Wang, D., Jain, A.K., 2018. Clustering millions of faces by identity. IEEE Transactions on Pattern Analysis and Machine Intelligence 40, 289–303. doi:10.1109/TPAMI.2017.2679100.

Pitolli, G., Laurenza, G., Aniello, L., Querzoni, L., Baldoni, R., 2020. Malfamaware: automatic family identification and malware classification through online clustering. International Journal of Information Security doi:10.1007/s10207-020-00509-4.

Schroff, F., Kalenichenko, D., Philbin, J., 2015. Facenet: A unified embedding for face recognition and clustering, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 815–823.

Sculley, D., 2010. Web-scale k-means clustering, in: Proceedings of the 19th International Conference on World Wide Web, Association for Computing Machinery, New York, NY, USA. p. 1177–1178. URL: https://doi.org/10.1145/1772690.1772862, doi:10.1145/1772690.1772862.

Shi, J., Malik, J., 2000. Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 22, 888–905.

Shi, Y., Otto, C., Jain, A.K., 2018. Face clustering: Representation and pairwise constraints. IEEE Transactions on Information Forensics and Security 13, 1626–1640. doi:10.1109/TIFS.2018.2796999.

Sibson, R., 1973. SLINK: An optimally efficient algorithm for the single-link cluster method. The Computer Journal 16, 30–34. URL: https://doi.org/10.1093/comjnl/16.1.30, doi:10.1093/comjnl/16.1.30.

Tapaswi, M., Law, M.T., Fidler, S., 2019. Video face clustering with unknown number of clusters, in: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).

Tian, G., Xia, Y., Zhang, Y., Feng, D., 2011. Hybrid genetic and variational expectation-maximization algorithm for gaussian-mixture-model-based brain mr image segmentation. IEEE Transactions on Information Technology in Biomedicine 15, 373–380.

Uykan, Z., 2021. Fusion of centroid-based clustering with graph clustering: An expectation-maximization-based hybrid clustering. IEEE Transactions on Neural Networks and Learning Systems , 1–15doi:10.1109/TNNLS.2021.3121224.

Vila, J.P., Schniter, P., 2013. Expectation-maximization gaussian-mixture approximate message passing. IEEE Transactions on Signal Processing 61, 4658–4672.

Wang, H., Wang, Y., Zhou, Z., Ji, X., Gong, D., Zhou, J., Li, Z., Liu, W., 2018. Cosface: Large margin cosine loss for deep face recognition, in: 2018 IEEE/CVF Conf. on Computer Vision and Pattern Recognition, pp. 5265–

5274. doi:10.1109/CVPR.2018.00552.

Wang, X., Imura, J., 2019. A gaussian process-based incremental neural network for online clustering, in: 2019 IEEE International Conference on Smart Cloud (SmartCloud), pp. 143–148.

Wang, Z., Zheng, L., Li, Y., Wang, S., 2019. Linkage based face clustering via graph convolution network. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) , 1117–1125.

Whitelam, C., Taborsky, E., Blanton, A., Maze, B., Adams, J., Miller, T., Kalka, N., Jain, A.K., Duncan, J.A., Allen, K., Cheney, J., Grother, P., 2017. Iarpa janus benchmark-b face dataset, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 592–600. doi:10.1109/CVPRW.2017.87.

Yang, L., Chen, D., Zhan, X., Zhao, R., Loy, C.C., Lin, D., 2020. Learning to cluster faces via confidence and connectivity estimation, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 13366–13375. doi:10.1109/CVPR42600.2020.01338.

Yang, L., Zhan, X., Chen, D., Yan, J., Loy, C.C., Lin, D., 2019. Learning to cluster faces on an affinity graph, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2293–2301. doi:10.1109/CVPR.2019.00240.

Yi, D., Lei, Z., Liao, S., Li, S., 2014. Learning face representation from scratch. ArXiv abs/1411.7923.

Yimyam, W., Pinthong, T., Chumuang, N., Ketcham, M., 2018. Face detection criminals through cctv cameras, in: 2018 14th International Conference on Signal-Image Technology Internet-Based Systems (SITIS), pp. 351–357.

Yin, J., Wang, J., 2016. A text clustering algorithm using an online clustering scheme for initialization, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, New York, NY, USA. p. 1995–2004. URL: https://doi.org/10.1145/2939672.2939841, doi:10.1145/2939672.2939841.

Yizong Cheng, 1995. Mean shift, mode seeking, and clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence 17, 790–799. doi:10.1109/34.400568.

Zhan, X., Liu, Z., Yan, J., Lin, D., Loy, C.C., 2018. Consensus-driven propagation in massive unlabeled data for face recognition, in: Proceedings of the European Conference on Computer Vision (ECCV).

Zhan, X., Xie, J., Liu, Z., Ong, Y.S., Loy, C.C., 2020. Online deep clustering for unsupervised representation learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

Zhang, L., Baek, J., 2019. Mixtures of gaussian copula factor analyzers for clustering high dimensional data. Journal of the Korean Statistical Society 48. doi:10.1016/j.jkss.2018.12.001.

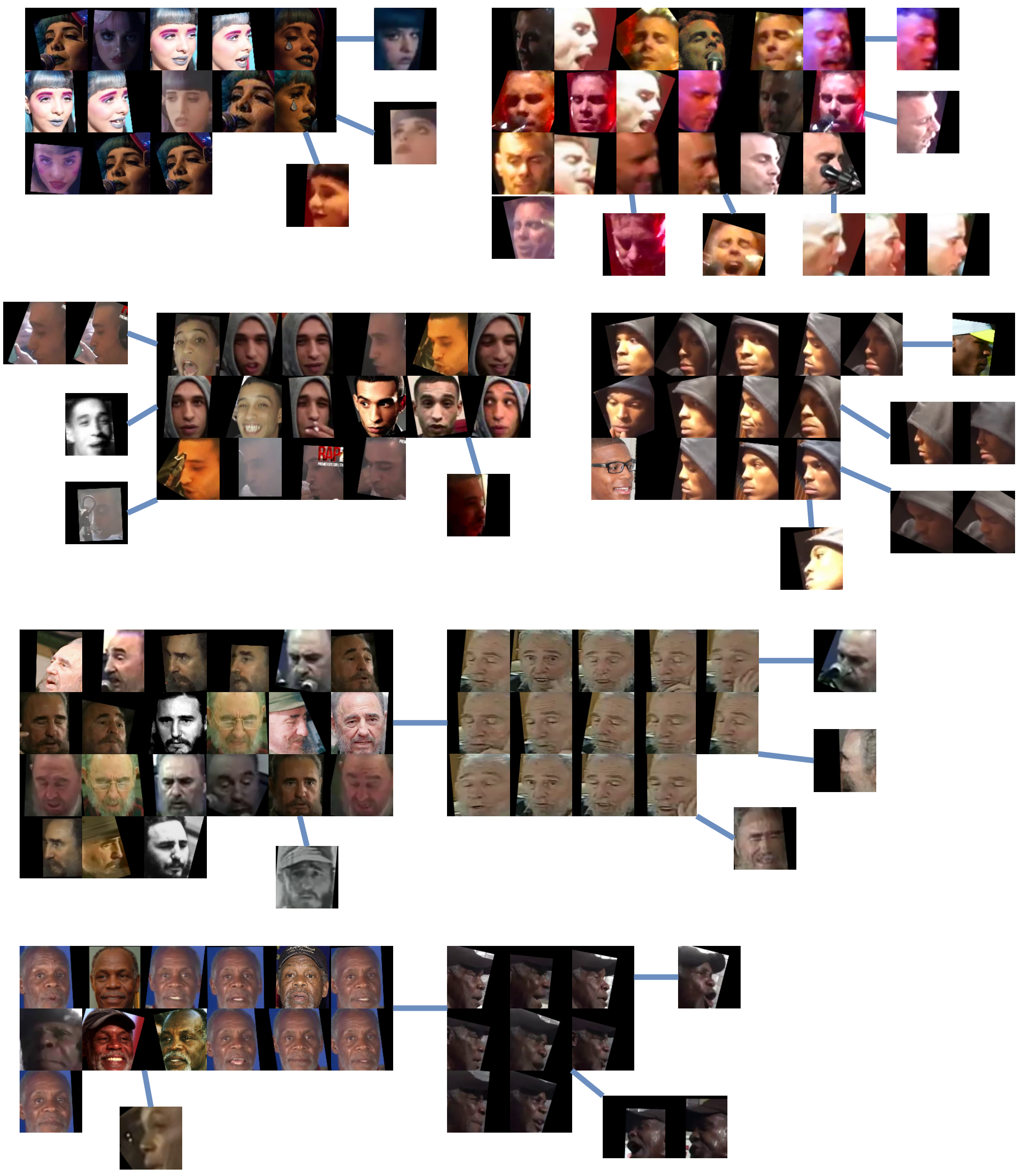


Fig. 11. Example clusters and connections generated by the proposed method in the IJB-C + 3 millions of distractors from VGG2 experiment. Each group of contiguous faces represents a cluster and each line represents a connection.

4.2 Learning to Automatically Catch Potholes in Worldwide Road Scene Images

- **Authors:** J. Javier Yebes, David Montero and Ignacio Arriola
- **Journal:** Intelligent Transportation Systems Magazine
- **Volume:** 13
- **Pages:** 192–205
- **Year:** 2021
- **Publisher:** IEEE

Learning to automatically catch Potholes in worldwide road scene images

J. Javier Yebes, David Montero and Ignacio Arriola

***Abstract*—Among several road hazards that are present in any paved way in the world, potholes are one of the most annoying and involving higher maintenance costs. There is an increasing interest on the automated detection of these hazards enabled by technological and research progress. Our work tackled the challenge of pothole detection from images of real world road scenes. The main novelty resides on the application of latest progress in Artificial Intelligence to learn the visual appearance of potholes. We built a large dataset of images with pothole annotations. They contained road scenes from different cities in the world, taken with different cameras, vehicles and viewpoints under varied environmental conditions. Then, we fine-tuned four different object detection models based on Deep Neural Networks. We achieved mean average precision above 75% and we used the pothole detector on the Nvidia DrivePX2 platform running at 5-6 frames per second. Moreover, it was deployed on a real vehicle driving at speeds below 60 km/h to notify the detected potholes to a given Internet of Things platform as part of AUTOPILOT H2020 project.**

***Index Terms*—Road Potholes, Deep Learning, Autonomous Vehicles, Internet of Things**

I. Introduction

ROAD DEFECTS are inherent to any road in the world due to several reasons such as weather, high traffic load and heavy vehicles. In some cases, large investments in infrastructure were done a long time ago and road surfaces have become more prone to deterioration such that they require frequent inspection and maintenance [1]. Other places around the world present different limitations and environmental conditions that influence the quality and state of the pavement [2] [3]. As a matter of fact, the road network is a valuable asset in any country because it serves the increasing demand of transportation for goods and people [4]. Consequently, an important budget is reserved for road reconstruction. For instance, Spain assigns an averaged 60% of its road maintenance budget to surface rehabilitation [5].

Commonly, road inspection has been carried out by qualified maintenance staff who drives along the road network, also stopping at several locations, to monitor and report encountered road hazards. However, automating these monitoring tasks can improve the safety of the staff and the effective detection of the hazards. In the recent years, technological progress has seen the installation of sensors and specialized equipment in maintenance vans, e.g. for the detection of road cracks [6]. Additionally, the progress in ADAS systems and the large investment in autonomous driving technologies have enabled the integration of multiple sensors in cars and the multi-modal perception of the environment [7]. In parallel, the advances in Artificial Intelligence (AI) [8] [9] have procured an effective use of the collected data from the sensors.

Among several types of road hazards (cracks, rutting, deteriorated markings, etc.) the target of our work is the automated detection of road potholes. They are present in worldwide roads causing discomfort to drivers, damages to vehicles and non-negligible repairing costs to public and private roads. For instance, the Department of Transport in UK stated in 2014 that more than £3 billion were spent nationally on road repairs. Also, the Royal Automotive Club (RAC) in UK estimated that vehicle repairing costs were around £100 million for all affected motorists and general surveys estimated that American drivers pay an average annual cost of $300 each to fix car damage due to potholes. Besides, specific funding has been provided to research on durable pothole repairs [10]. In fact, recent harsh winters and springs, weather changes and transportation demands are causing a rapid increase of road potholes.

Our research has received funds from the AUTOPILOT H2020 project [11], which will deploy, test and demonstrate automated services based on Internet of Things (IoT) in five driving modes. In the Highway Pilot use case, a cloud service merges the sensors' measurements from different IoT devices to locate and characterize road hazards. The goal is to provide the following vehicles with meaningful warnings and driving recommendations to manage the hazards in a safer or more pleasant way. For a better understanding of the scenario, we assume the following: Firstly, a vehicle equipped with different systems has the role of IoT device, which is comparable to smartphones and other wearables that can send/receive messages to/from IoT platforms. Secondly, in-vehicle systems include AI modules that process data and produce low-bandwidth messages that are wirelessly sent to IoT platforms.

Within this background, our main research motivation is the automated visual detection of road potholes from a frontal colour camera on board a vehicle. Once potholes are detected, their location is reported to a given IoT platform. Current state of the art is predominantly based either on sensing potholes with accelerometers [12] or cameras [3]. Accelerometer-based detection requires that the vehicle drives

J. Javier Yebes, David Montero and Ignacio Arriola are with the Department of Intelligent Transport Systems and Engineering, Vicomtech, Paseo Mikeletegi 57, 20009 Donostia/San Sebastián, Spain (e-mail: jyebes; dmontero; iarriola@vicomtech.org).

over the potholes, which is usually not the case as the driver will try to avoid them. Vision-based detection is naturally seen as the same process in which drivers perceive the environment and anticipate to possible road hazards. For the latter one, classical image processing and machine learning approaches have been evaluated on images of certain world regions. Our strategy is to automatically learn the visual appearance of worldwide potholes using latest advances in AI, i.e. Deep Neural Networks.

Therefore, the contributions of this paper are the following:

- We **built a dataset** with manual annotations from several **places around the world** that include images from Europe, America, Asia and Africa. It is composed of challenging scenes captured from different cameras, viewpoints and under varied environmental conditions.
- We fine-tuned and evaluated 4 different Deep Neural Networks **(DNNs)** for the **detection of road potholes on images**. We achieved high detection ratios ($mAP > 75\%$) considering the high intraclass variance.
- The pothole detector was **tested on the Nvidia DrivePX2** platform for embedding ADAS in driverless vehicles.
- As part of AUTOPILOT project, the pothole detector was **integrated on a real vehicle**. The set-up included an automotive grade camera, General-Purpose computing on Graphics Processing Units (GPGPU) and IoT communication modules on board the vehicle.

The remaining of the paper is organized as follows: In Section II we review sensing modalities and literature related to the detection of road potholes. Next, Section III presents the AUTOPILOT European project and related research areas. Section IV describes our approach to learn the visual appearance of potholes with Deep Neural Networks and Section V explains the created dataset, the experiments carried out and the obtained results. Finally, Section VI concludes the paper.

II. Related work

The inspection of roads for surface damage is a worldwide challenge. The literature reviewed in this section includes systems from places all around the globe. Commonly, qualified maintenance staff monitors and reports encountered road hazards. In the recent years, technological progress has enabled the installation of specialized equipment in maintenance vans [6]. Nowadays, smartphones are readily available and they can be used to take images on the road from any vehicle [1] [13].

The remaining of this section is dedicated to detection approaches for our targeted hazards: the road potholes. Attending to sensor modality, there are four categories: *ultrasonic*, *accelerometer*, *image* and *combination of image and depth*.

A. Ultrasonic-based pothole detectors

The system in [14] consisted on a prototype robot vehicle with one ultrasonic sensor and a Zigbee communication module. It was designed to detect potholes with a minimum depth of 1 inch and to broadcast warning messages to vehicles in its vicinity (100m). However, it was tested as a lab prototype without real data. Another system was attached to a motorbike and tested on Indian roads [2] in which the ultrasonic sensor was used to determine the depth and height of potholes and speed humps. However, motorbikes are easy to manoeuvre and drivers will typically try to avoid driving over the hazards, thus reducing its applicability.

B. Accelerometer-based pothole detectors

Recently, many studies have explored accelerometer signals from smartphone sensors and on board vehicle sensors. Mobile sensing was researched in [15] that addressed the differences between sensors embedded in 4 distinct phones and compared various z-axis thresholding algorithms for the detection of potholes. Similarly, [16] presented a real-time pothole detector in which the signals were normalized to account for smartphone position and attitude. However, a single pothole case study of size $51 \times 58 \times 6$cm was used.

Given the limitation of simple heuristics and thresholding, Machine Learning (ML) became into scene. Bhatt et. al. [12] presented an intelligent system in a smartphone where the pothole detections (at 5HZ) were marked in a map. They concluded that Support Vector Machines yielded the best accuracy in the classification of road conditions. Alternatively, the PADS [17] system was based on K-Means, tuned thresholds and a tri-axial accelerometer on board the vehicle (no smartphone). PADS solution was based on IoT, sending detections to a remote server and marking their approximate location in Google Maps. A different approach was introduced by Fox et. al. [18] for the aggregation and cloud processing of crowd-sourced data from inertial sensors on board vehicles.

As discussed in [17], the accelerometer-based approaches present several drawbacks. Vehicle cushioning mitigates the vibrations produced by potholes, experiments become biased and the system can be confused due to braking and other road anomalies. Moreover, smartphone-based solutions depend on model and type of mounted accelerometers and their location and orientation in the vehicle. Achieving a self-calibration and a correct alignment of axes is not straight-forward [19]. Furthermore, these systems assume that at least one of the vehicle wheels passes over the pothole which may not happen if the driver slightly swerves to avoid them or if the potholes are in the center of the lane.

C. Image-based pothole detectors

A naturalistic approach is to perceive the road scene with cameras and analyse the images in search of potholes, which mimics the visual inspection of humans. One of the early research works consisted on road condition assessment using hyperspectral aerial imagery [20].

More recently, two works applied simple image processing techniques for pothole segmentation. [21] collected grayscale images from a camera on board a vehicle and [22] employed colour images from Google search engine using the keyword ”pothole”. However, both conducted few tests and obtained non-sufficient evidence of the accuracy.

A key reference was [3], which enabled an initial analysis of the vision-based pothole detection challenge. They released a publicly available and annotated dataset of varied road potholes taken with a camera attached to the vehicle windscreen, driving at normal speed ($< 40Km/h$) and under various illumination conditions. Further details of the dataset are provided in Section V. The image processing techniques in [3] were limited compared to the state of the art in AI. Although reported results were promising, they depended upon several computer vision filters with parameter tuning. It must be observed that the continuation work [23] evaluated computer vision algorithms for pothole distance estimation but this task is out of the scope of our paper.

In [24], a spatio-temporal saliency map was proposed to detect potholes in road scenes with heavy traffic. Grayscale images from a dashcam were used for the purpose and their visual properties were studied. [25] applied ML methods to classify images in two sets as pothole/non-pothole. They employed HOG features and Naive Bayes classifier. Besides, they combined the approach with normalized graph-cut segmentation to locate pothole regions on the positively detected images. For the evaluation, they relied on a private dataset of limited size and reported an elapsed time of 0.7s per image containing potholes.

Additionally, some newer proposals used images captured by a smartphone. In [13], superpixels were computed from grayscale images and the texture of the scene was analysed with wavelets. Then, a set of subtractions were applied to identify anomaly image patches that were marked as belonging to pothole category. Although the reported accuracy was high, there was a lack of thorough analysis over the dataset. In contrast to our evaluation on vehicle hardware platforms, their system was implemented on a high-end desktop computer. A similar hardware was used in [26] with the addition of a Nvidia GPU 770M to run different Deep Neural Networks (DNN) that classified images in two categories: one or more potholes in the image vs no potholes. All models reported high accuracy above 96%. However, the models were trained in a binary classification task without identifying the box around the pothole. Our DNN system does provide the detected box.

With regards to Deep Learning approaches, Maeda et. al. [1] has recently presented a road damage detection using DNNs. It defined 5 classes of road cracks and 3 mixed categories that gathered other types of hazards (rutting, bump, pothole, blurred markings). An image dataset with 15,435 annotations was built while driving in 7 cities of Japan. During our evaluation, we extracted images and labels from their dataset for comparison. However, their work was mainly focused on road cracks. A further review is provided in Section V.

D. Depth- and vision-based pothole detectors

The addition of depth data to the 2D visual clues is also present in the literature. [27] evaluated statically on the street the use of the Microsoft Kinect camera. However, this sensor is not well-suited for outdoors due to the InfraRed structured lighting that is mostly overridden under the presence of ambient IR from the sun.

Stereo vision for pothole detection was firstly introduced in [28]. Three phases were proposed to construct the disparity map, fit a surface and apply a fixed height threshold to detect shallow road regions which potentially belonged to potholes. Stereo cameras were also employed in [29], which implemented a real-time pothole detector on a digital signal processing unit based on a very similar algorithm. It must be noted that these approaches are sensitive to disparity estimation errors that would yield false pothole detections. In fact, [29] required a stereo pair of cameras pointing downwards to the road. This is contrary to the tendency of placing cameras with XZ-plane nearly parallel to the ground such that other tasks can be done with the same images, e.g detection of objects/lanes ahead of the vehicle.

In [30] a 2D LiDAR and a camera were combined. Different pothole detection algorithms for each sensor modality were presented and the experiments were carried out in lab placing a shallow box to simulate the pothole. Thus, the setup was far from any real road scene. However, [31] used real data and introduced the detection of speed bumps on LiDAR 3D data at the same time potholes were detected with basic image processing techniques.

To complete the literature review, there exist other private approaches without further technical details available: a Jaguar Land Rover warning system [32], a Google patent [33] and a commercial black box camera for pothole detection [34]. In the latter one, size, location and appearance of potholes were sent to a centralized server for off-line assessment and maintenance actions. The system was only tested on sunny weather.

Compared to above mentioned works, our proposal used an automotive grade camera installed on the front bumper of a car that can be deployed in new manufactured vehicles. Moreover, we applied latest progress in Deep Neural Networks (DNN) and tested the pothole detector in an embedded driverless vehicle platform. We built an image dataset with road scenes from different camera sources under varied environmental conditions to train the DNN models. Moreover, as part of the AUTOPILOT project [11], the detected potholes were reported in IoT-fashion to cloud services. The following section contextualizes our work within that project and 3 interrelated research areas.

III. AUTOPILOT AND RELATED RESEARCH AREAS

In the last 30 years, there has been a huge progress in the field of Autonomous Vehicles (AV) in such extent that nowadays, there exist vehicles in the market that claim several autonomous driving functions and many more yet to come. From a computer vision perspective, [7] provides a wide literature review.

Successful approaches in Artificial Intelligence (AI) have contributed to the progress. They have been proposed either as knowledge transfer from other research areas or as specific solutions for driver assistance an driverless vehicles. Indeed, AI is a broad concept referring to the cognitive ability of machines as opposed to the natural intelligence of humans or other beings. In recent history, three terms have become very popular which represent different trends in the algorithmic implementation of AI. In chronological order: *Pattern Recognition (PR)* [8], *Machine Learning (ML)* [35] and *Deep Learning (DL)* [9]. PR can be considered outdated but it belongs to the age when the premier international conference on Computer Vision and Pattern Recognition (CVPR) was born. ML is still widely employed offering supervised and unsupervised learning methods. However, it has been recently outperformed by DL in many tasks and challenges at the cost of higher computational requirements and dedicated hardware.

In parallel, Internet of Things (IoT) extends the principle of internet services to include all possible types of devices, thus bringing the advance in accessibility of information from the virtual to the physical world. IoT can enable new capabilities in AV for various layers. From the communication for a given cooperation zone within vehicle range to the connection to cloud platforms with the aim of sharing information in larger zones, i.e. smart cities.

In 2016, the European Commission funded five Large-Scale Pilots (LSPs) on the IoT. The **AUTOPILOT** H2020 project [11] was selected as Pilot 5: *autonomous vehicle in a connected environment*. AUTOPILOT concerns the use of IoT for enabling automated driving as it is illustrated in Fig. 1.

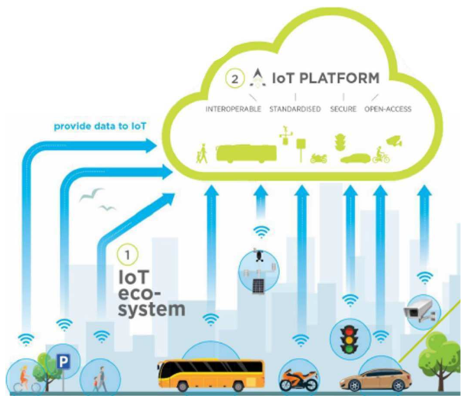


Fig. 1. AUTOPILOT H2020 project illustration [11]

AUTOPILOT will deploy, test and demonstrate IoT-based automated services in five driving modes. The research work presented in this paper was applied to the Highway Pilot use case to locate and characterize road hazards like road surface anomalies, bumps, fallen objects, etc. The fusion of advances in AI, AV and IoT are studied to provide feasible solutions as the one proposed in this paper.

In AUTOPILOT, a cloud service merges the sensors' measurements from different IoT devices on board vehicles (camera, LiDAR, inertial and vibration sensors) and on the road side (mainly cameras). The goal is to provide incoming vehicles with meaningful warnings and driving recommendations to manage the hazards in a safer or more pleasant way. Moreover, the automated detection of the hazards enables the reporting to corresponding traffic or infrastructure authorities for maintenance or other monitoring and controlling actions.

Considering this background, our current research work has a direct impact on the operational performance of the new generation of ITS. One the one hand, recent progress in AI, more specifically Deep Learning, enables the integration of high-performing scene perception modules into vehicles equipped with multiple sensors. On the other hand, IoT enables the data collection from multiple sources to also implement AI solutions on the cloud. The following section describes the details of our pothole detector model based on latest AI.

IV. Learning pothole appearance with DNNs

Definition of potholes: Bowl-shaped holes of various sizes in the pavement surface that are greater than $175cm^2$ in area ($\sim 15cm$ diameter) (illustrative examples in Fig. 2).

A set of factors generate road distress that leads to the formation of potholes [23]. The most relevant causes are the volume of traffic, the axle load of heavy vehicles (buses and trucks) and the environmental conditions (day/night temperature contrast, snow, rain). Moreover, the type of road surface material, the underlying terrain and construction techniques influence the quality and resistance of the pavement.

Typically, the first phase is the formation of cracks that allow water to seep through. Next, when vehicles drive over them, the water is pushed in many directions around the initial cracks. Eventually, cavities will appear in the asphalt growing in size until they reach pothole dimensions. Besides, the wider the cavity the quicker the process. Therefore, automated detection and notification of potholes helps preventing progressive road deterioration. Our research work proposes to learn the visual appearance of potholes with Deep Neural Networks.

A. *Visual appearance of potholes*

We have collected images with several types of potholes under different illumination and weather conditions as depicted in Fig. 2. The most common potholes show a pronounced edge describing an elliptical shape. However, there are some that describe a more square-like shape and some others that do not have a pronounced edge. They might appear darker or brighter on the background pavement. The deepest ones look darker because of the shadow of the edge, while in the flat ones it is possible to see the ground or gravel inside the pothole. Besides, some potholes appear filled with water and might also reflect surrounding scene in the surface.

Attending to this intra-class variance, pothole detection based on classic computer vision has limitations. The literature reviewed in Section II-C presented ad-hoc image processing filters and heuristics, which were biased towards specific datasets and conditions. We propose to take successful DNN object detection models as a base, thus to automatically learn more discriminative visual features of the potholes in varied scenes.

B. *Deep Learning for the detection of potholes*

Object detection approaches based on engineered features and Machine Learning classifiers have been very fruitful until recent times [35]. However, when applied to different tasks or adapted for additional challenges they required intensive parameter tweaking and dimensionality reduction [36].

Recent trends on Deep Learning have achieved impressive detection performance on various tasks [9] including object

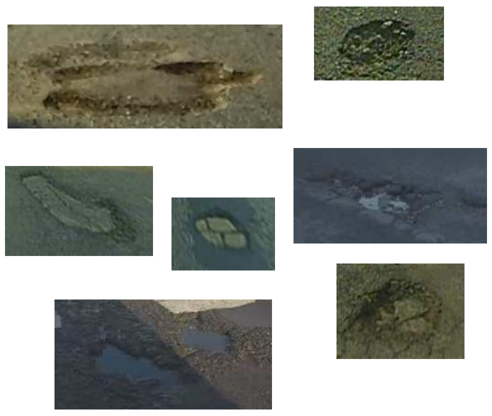


Fig. 2. Samples of annotated potholes. They come from several sources and represent varied places, environmental conditions and camera viewpoints.

detection [37]. Moreover, several DNN based models are publicly shared and can be used as initialization step for further training and fine-tunning in different object classification tasks [38]. We make use of this progress in AI for the detection of potholes.

Based on the investigation in [37] and the entries reported in the TensorFlow model Zoo [38], we selected 4 models that have shown high detection ratios at reasonable processing costs. Attending to detection performance we chose 3 different configurations of the architecture *Faster R-CNN*. Besides, we evaluated the Single Shot multibox Detector (*SSD*) because it is targeted for mobile applications. They are explained in the next paragraphs.

The Faster R-CNN architecture uses a system called Region-based Convolutional Neural Network (R-CNN). As opposed to a brute-force sliding window approach over image space, regions of interest are proposed and warped to fixed size and then, they are individually fed into a CNN for classification and bounding box refinement. The Faster R-CNN network architecture consists of three major blocks (see Fig.3):

1) A CNN to extract features. In brief, the weights of several convolutional filters are learned in order to calculate the feature maps.
2) A Region Proposal Network (RPN) that acts as attention model. It proposes regions that may contain an object. During training, it learns to determine whether a certain region has an object or not and the appropriate shapes and sizes. This is the multi-task loss:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i,p_i^*)+ \\ +\lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(\boldsymbol{t}_i,\boldsymbol{t}_i^*) \quad (1)$$

where i is the ith candidate region (called anchor in [39]), p_i is the predicted probability to be an object, p_i^* is 1 if the candidate truly contains an object and 0 otherwise. $\boldsymbol{t}_i$ are the coordinates of the candidate region, $\boldsymbol{t}_i^*$ is the ground-truth box, L_{cls} is the log loss over two classes (object/non object) and L_{reg} the regression defined as the robust smooth L1 function [39].
3) The last block is the object classifier Fast R-CNN [40] that receives the proposed regions, assigns a class to them and it also refines the bounding boxes using a regressor. The loss function in training is very similar to the one in the RPN.

We have fine-tuned these blocks end-to-end using stochastic gradient descent optimization with back propagation.

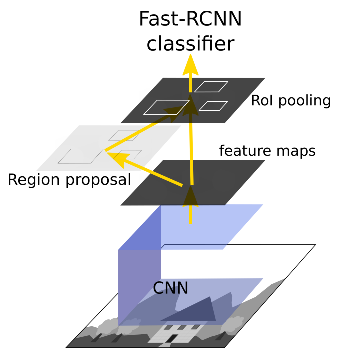


Fig. 3. Simplified diagram of Faster R-CNN for illustration purposes

The seminal method was very slow to train because of the computation of features for every region. Then it eventually evolved into Faster R-CNN [39]. Despite its ”Faster” prefix, it is slower than SSD because it re-runs the cropped regions through the feature extractor. In fact, we selected three Faster R-CNN models that differ in the feature extractor because the choice of feature extractor has a higher impact in Faster R-CNN models than in SSD. The choices are introduced below:

Faster R-CNN Inception v2. The feature extractor in this approach provides batch normalization for accelerating the model training and it has yielded high accuracy [41]. Compared to v1, it is more efficient by factorizing the convolution operations. Also, it is a wider network to avoid losing visual details that may happen in v1 which is deeper as road potholes typically represent small portions of the images.

Faster R-CNN Resnet101. This model training uses Resnet101 that stands for Residual Network with 101 layers [41] and has achieved high success on many competitions. This type of networks try to learn residuals which are short-cut connections between layers. The approach allows to train deeper models without degradation.

Faster R-CNN Inception-Resnet v2 (atrous). This third model uses a hybrid feature extractor that combines Inception and Resnet, second version (v2), yielding improved recognition performance as reported in [42]. Moreover, the ”a trous” (*with holes in*) option employs dilated convolutions, which provide a wider field of view at the same computational cost towards achieving better accuracies.

Opposed to above ones, the Single Shot multibox Detector (SSD) uses a single feed-forward convolutional network [43].

It is a faster and simpler method because it discards the proposal generation phase and feature re-sampling. It uses a set of default boxes with different aspect ratios and scales per each feature map location. These boxes are equivalent to anchors in Faster R-CNN and represent priors manually chosen as described in [43]. For the sake of clarity, the training loss is reproduced below. It is a combination of two functions that capture object confidence and location error:

$$L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + L_{loc}(x,l,g)), \qquad (2)$$

where N is the number of matched default boxes and x is 1 to mark detections correctly matched to ground-truth or zero otherwise. L_{conf} is the softmax loss over classes confidences (c) and the localization loss L_{loc} is a smooth L1 function between predicted (l) and ground-truth box (g) parameters.

SSD Mobilenet v2. We selected *Mobilenet version 2* as feature extraction network [41], which has been specifically designed for mobile vision applications with limited resources. It uses depth-wise separable convolutions and the version2 incorporates a set of improvements. One of them are residual connections as in ResNet but applying thinner bottleneck layers [44] which makes it more efficient than v1.

C. Fine-tune DNN models for the detection of road potholes

The selected models were successfully evaluated on multi-object detection challenges and we fine-tuned them for a single class: road potholes. We downloaded the above mentioned pre-trained DNN models [38], which are used as initialization. The weights of the feature extractor layers are automatically fine-tuned during the training. The weights of the remaining layers that perform the classification and object localization tasks, are randomly initialized and learned towards the visual detection of potholes. This is a established practice when applying Deep Learning models. As opposed to hand-engineered features such as SIFT, SURF, HOG which are well-known in computer vision literature, these DNN models use automatically learned features from the database *Common Objects in Context (COCO)* [45]. This large-scale object detection dataset contains more than 200K images distributed in 90 object categories depicting real world scenes. It is sufficiently large to learn rich visual features of naturalistic scenes that are encoded in the weights of the neural networks. These features cannot be learned using a limited number of object labels, like our pothole case. Hence, the pre-trained models transfer the knowledge in order to train the pothole detector.

For the fine-tuning of the models, we modified several parameters after conducting an analysis on the dataset. In the following experimental section, we first described the dataset and then, the details of the changed parameters in Section V-B.

V. Experimental section

A. Dataset

The images used in our research have been obtained from varied sources and belong to different places in the world. Only a subset was already labelled and several images have been manually annotated with boxes around potholes. Our motivation to build this dataset has been twofold: I) gathering enough data samples for fine tunning and evaluation and II) increase variance in pothole appearance, camera viewpoints and road scenes. We have divided the images in training, validation and test sets.

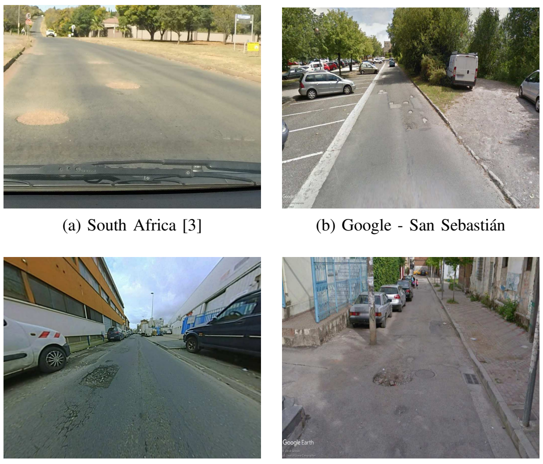
*1) **Training and validation sets**:* The database consists of 5,874 images from which 5,774 have been used for fine-tunning and 100 have been randomly picked for validation. In the training set there is a total amount of 9,716 potholes while in the validation set the number of potholes is 171. All the images have been captured from a vehicle, with different camera placements and viewpoints. We constructed the training set from these sources:

- 4,030 images of size 3680×2760 pixels and their labels were obtained from [3]. The images were captured as regular snapshots from a GoPro camera attached to the inner side of a vehicle windscreen.
- 1,644 images have been sampled from AUTOPILOT videos of VALEO in the surroundings of Paris and potholes have been manually labelled. These images have a resolution of 1280×800 pixels. The sequences were captured by an automotive-grade fish-eye camera mounted on the front bumper of a Volkswagen Tiguan 2. We removed the radial distortion as pre-processing step using the calibration parameters of the camera. Then, the undistorted images were added to the dataset.
- The remaining 100 images (up to 5,874) have been captured from the Google Earth Pro street-view tool at a resolution of 1236×804 pixels. They have been also manually labelled. Most of these images containing potholes belong to streets of Tirana in Albania and some have been found in San Sebastian and Bilbao in Spain.

We used the tool *Label Img* [46] to annotate the boxes enclosing road potholes. The images were slightly cropped near the borders to remove some scene background and resized to 1024×800 pixels (see Fig. 4). The motivation for this size is explained in Section V-B.

We analysed the distributions of the aspect ratio and area of the annotated potholes for fine-tunning the training parameters of the DNNs. We provide box plots in Fig. 5. The size and location of the blue box indicates those values from the first (25%) to the third quartiles (75%) and the interquartile distance ($IQR = Q_3 - Q_1$). The line inside the box represents the median value. The whiskers represent the highest observation below the upper limit ($L_{upper} = Q_3 + 1.5 \cdot IQR$), and the lowest observation above the lower limit ($L_{lower} = Q_1 - 1.5 \cdot IQR$). The points outside those two limits are considered outliers.

*2) **Test set**:* The test images have been collected from different sources as well. In first place, some additional images have been fetched from the sequences recorded in AUTOPILOT project with VALEO vehicle. The videos were captured in different dates and environmental conditions compared to the training set. The routes were also nearby Paris sharing some streets with the training sequences. Indeed, we manually selected 265 images with 128 potholes in challenging scenarios, which included dark lighting (tunnels), rough


(a) South Africa [3] (b) Google - San Sebastián

(c) Valeo - Paris (d) Google - Tirana

Fig. 4. Sample images of the constructed training+validation image set.

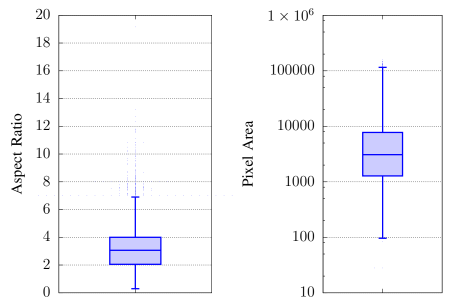


Fig. 5. Distribution of the aspect ratio and area in pixels of the annotated potholes for the train dataset. The boxplot on the left has a linear representation of the aspect ratios. The quartiles are $Q_1 = 2.053, median = 3.062, Q_3 = 4.0$. The boxplot on the right shows the area in pixels and has a logarithmic axis for the purpose of visualization. The quartiles are $Q_1 = 1276, median = 3081, Q_3 = 7744$ pixels

weather conditions, traffic jams and roads with bumps, stains and manholes. The latter ones being potential cases of false positive detections as potholes.

The second source is a youtube video[1] recorded from a dash-cam in a car driving through a road stretch in the area of Willamette National Forest, Oregon, US. We extracted 482 frames and manually labelled 475 potholes.

Additionally, we re-annotated some of the images in [1]. The road damage inspection presented in that paper released a public dataset with 9,892 pictures captured with smartphones. Among the classes of hazards identified in [1], one was categorized as *Rutting, bump, pothole and separation*. It is a very broad class with loose large boxes around the hazards. Hence, we reviewed them to generate new labels only for potholes discarding the remaining annotations. As a result, 62 potholes have been selected within 54 images.

[1]https://www.youtube.com/watch?v=BQo87tGRM74

B. Experimental details of DNN models of road potholes

As introduced in Section IV-C, we fine-tuned 4 DNN models for the detection of road potholes. In first place, we studied the nature of the dataset and we fixed the size of the input layer to 1024×800 pixels. We found that the median area of potholes corresponded to a 0.37% of the original image size. Given a reduced resolution of 600×600 as reported in [38], $1350 \simeq 36^2$ pixels is the average area of the potholes in the images. Besides, the 1:1 aspect ratio involves warping, cropping and resizing the original images. Consequently, road areas on the left and right sides in front of the car are cropped out and the visual appearance of the scene and the potholes are deformed and reduced in granularity. Therefore, we decided to use a larger window size with an aspect ratio similar to the original resolution of the images, achieving a median area of 3081 pixels among the annotated potholes.

In addition, we applied a set of adjustments for fine-tuning the 3 Faster R-CNN models.

- Added the aspect ratios of 1:3 and 1:4 after analysing the dataset (see Fig. 5). Thus, the aspect ratios during learning were $[.5, 1, 2, 3, 4]$.
- Reduced the maximum number of region proposals for Faster R-CNN models from 300 to 100, with the aim of decreasing detection time without losing performance [37].
- Due to the limited size of our pothole database, we increased the number of training samples by data augmentation. Among several options for fine-tunning Faster R-CNN models, we selected *random_adjust_brightness* and *random_horizontal_flip* because of their effectiveness in previous research experience for object detection.
- We enabled the drop-out feature in the second stage of the Faster R-CNN training to prevent over-fitting. Basically, this option randomly drop units and their connections from the network, which prevents the units from co-adapting too much.
- The number of steps employed for fine-tuning the 3 models was 2 millions. We picked this number observing the performance of the trained models on the validation subset. The loss function was stable without showing too much improvement, thus we decided to stop at 2M steps, which corresponds to 173 epochs.

Similarly, we applied the same adjustments for fine-tuning the *SSD Mobilenet v2* model but using a a squared size of 800 × 800 pixels.

C. Implementation details for training & evaluation

We include in this section the relevant hardware and software details of the experiments. For the fine-tuning we have used a barebone PC equipped with 2 × Xeon 20-Core @ 2.2GHz, 16 × 32GB RAM modules and 8 × GPGPU Nvidia Tesla P100M with 16GB of memory each one. It must be noted that only one GPGPU was used during the learning of a given DNN model. The software environment included Ubuntu OS, Tensorflow 1.7 with Tensorboard for the visualization and validation of the learning process [47].

For the evaluation of the models, we used the same PC and also the Nvidia DrivePX2 Autochauffeur[2], which is specifically designed for research and embedding AI solutions in driverless vehicles. This platform has two Nvidia Tegra X2 SoCs (known as TegraA and TegraB), where each SoC contains 2 Denver cores, 4 ARM A57 cores and a Pascal GPU. This replication is for redundancy purposes and not for parallelization. The dGPU has 1152 CUDA cores and 4GB of memory while the iGPU has 256 cores and uses the shared system memory 7GB. Besides, the device includes a set of interfaces (GMSL, USB, Ethernet, CAN, FlexRay, etc.) for different sensors like cameras, LiDAR, GPS, CAN bus signals, etc. The OS is based on Ubuntu Linux distribution and we carried out the tests with version *Drive 5.0.5.0a*, which includes CUDA9. Additionally, Tensorflow 1.7 was installed to load the learned DNN models and automatically find potholes in road scene images.

D. Results

In our research, we considered the *mean Average Precision* (mAP) as detection performance metric and the higher the mAP, the better. We analysed two different approaches to obtain it: COCO and PASCAL. Both of them count the number of true/false positives and false negatives to calculate precision and recall. Next, the Average Precision is obtained as

$$AP = \frac{1}{11} \sum_{r \in \{0,0.1,\ldots,1\}} \max_{\tilde{r}:\tilde{r} \geq r} p(\tilde{r}), \qquad (3)$$

where r and p are the recall and precision values respectively and 11 equally spaced sampled recall points are used.

To determine object localization accuracy, the Intersection over Union (IoU) measures the overlap between predicted and ground-truth bounding boxes around objects. For the estimation of mAP, COCO defines the *mean Average Precision* over all object classes and multiple values of IoU: precision-recall curves are evaluated at $IoU = [.5 : .95]$ in steps of 0.05. In PASCAL mAP, only one value of IoU=0.5 is reported. Then, using COCO strategy, models showing higher precision in object localization are rewarded. For our analysis, we have also decided to report PASCAL mAP @ IoU=0.4. Precisely annotating the location and boxes around potholes is challenging, mostly on road scenes with deteriorated surfaces or road defects that could be labelled as one or several potholes. Observing the results, we realized that many pothole detections on the road surface could be considered as good by a human viewer but they were classified as false positives or false negatives by the evaluation algorithm. Therefore, with the aim of reducing the number of FN and FP we also report the mAP @ IoU=0.4.

Before presenting our results, for reference, we reproduced here the mAP and processing speed of the pre-trained models as reported in COCO entries [38]. The timings were obtained for images of 600×600 pixels and running on a Nvidia GeForce GTX TITAN X.

1) ssd_mobilenet_v2_coco ($t = 31ms$, $mAP = 22$)
2) faster_rcnn_inception_v2_coco ($t = 58ms$, $mAP = 28$)
3) faster_rcnn_resnet101_coco ($t = 106ms$, $mAP = 32$)
4) faster_rcnn_inception_resnet_v2_atrous_coco ($t = 620ms$, $mAP = 37$)

Running time and detection performance are correlated: the higher mAP, the higher processing delay. The reason is due to neural network architecture complexity and number of layers, provided that the size of the input layer is constant for all of them. A thorough analysis can be found in [37].

Next, we report the results obtained after fine-tuning *SSD Mobilenet v2* for potholes and evaluating the model on the test set. The AP was 33.62% and 45.57% @ IoU 0.5 and 0.4, respectively and took approximately 455ms per image when executed in Nvidia DrivePX2. These values show a very low detection performance despite the fine-tunning of some parameters. We investigated the main reasons and reached the conclusion that in SSD Mobilenet the discretization in bounding boxes and their separate analysis does not account for neighbouring pixels, which provide useful context information. Despite the initial expectations of this research to implement a mobile embedded Deep Neural Network, the SSD Mobilenet is not well suited to detect objects that strongly depend on the appearance of surrounding scene, i.e. potholes vs road appearance. In addition, by design it does not cope well with small bounding boxes compared to the image size. However, our dataset contains small annotated boxes compared to the size of the image.

The remaining of this section is dedicated to the results of the selected Faster R-CNN models. Figure 6 presents the precision-recall curves on the test set for the 3 Faster R-CNN models at two different values of IoU and Table I shows precision and recall based on PASCAL evaluation metrics. Table II shows the mean Average Precision for each of them. Moreover, running times are shown in Table III. The model names have been shortened[3].

TABLE I
PRECISION (P) AND RECALL (R) OF THE EVALUATED POTHOLE DETECTORS BASED ON FASTER R-CNN MODELS.[3]

IoU	Model	p (%)	r (%)
0.5	inception_v2	70.10	65.85
	resnet101	**83.48**	**75.23**
	inception_resnet_v2_atrous	73.08	69.78
0.4	inception_v2	74.7	70.26
	resnet101	**82.27**	**82.16**
	inception_resnet_v2_atrous	76.5	73.07

As it can be observed from the resulting values, **Faster R-CNN Resnet101** yields the highest detection performance both in PASCAL and COCO evaluation metrics. It is not the slowest DNN but it requires on average 432.7ms per frame on the Nvidia DrivePX2. For the goals of detecting potholes and

[2]https://www.nvidia.com/en-us/self-driving-cars/drive-platform/

[3]For visualization purposes the prefix "Faster R-CNN" has been omitted

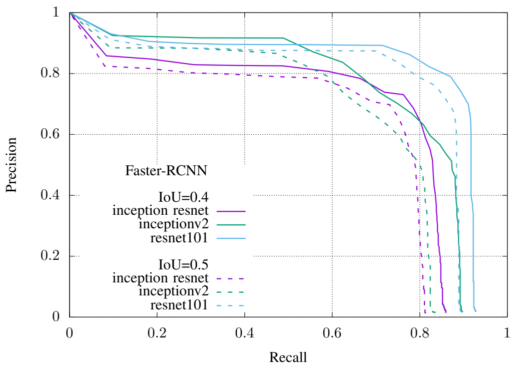


Fig. 6. Precision-recall curves for the test set. Three Faster R-CNN models with a different feature extractor network and two values of IoU are compared. Due to the nature of road potholes, there is a larger error in the localization accuracy of the ground-truth bounding boxes. Thus, we opted to report IoU = 0.4 in addition to the commonly used 0.5 value. The aim is reducing false negatives and false positives in road scenes with deteriorated surfaces where manual pothole labelling is also challenging.

TABLE II
MEAN AVERAGE PRECISION OF THE 3 POTHOLE DETECTORS.[3]

		mAP(ˆ1)	
IoU	**Model**	**PASCAL**	**COCO**
0.5	inception_v2	67.05	27.51
	resnet101	**77.49**	**31.12**
	inception_resnet_v2_atrous	68.71	26.67
0.4	inception_v2	75.45	—
	resnet101	**82.02**	—
	inception_resnet_v2_atrous	69.72	—

reporting them to an IoT platform (AUTOPILOT project), it is a valid time. There are not real-time requirements because detected potholes will be monitored from control centres and broadcasted to warn other road users when sufficient confidence is reached after several repeated detections on a given map location. For vehicle reactive manoeuvres upon detection, further research is needed to optimize the neural networks without losing too much performance in order to obtain processing gains. A significant reduction of size on the input layer could importantly increase the number of false detections due to the small size of potholes compared to the rest of the image. Thus, other DNN optimization strategies should be explored, like enlarging the dataset, retraining on mobile or pruned networks from scratch and transforming DNNs with software optimization tools for its execution on specific hardware.

Also, from the results, it is surprising that *Faster R-CNN Inception Resnet v2 (a trous)* obtains lower AP, which is contrary to expected performance. Indeed, comparing the AP @ IoU 0.5 and 0.4 it can be seen that a slight gain of 1% is achieved, meaning that this DNN obtains more false negatives and more false positives that do not overlap correctly to ground-truth potholes. Besides, the validation p-r curve during fine-tuning shows a lower performance than the other models. On the one hand, the complexity of this network and the limited number of training samples can cause weak propagation of relevant signals for the class potholes. On the other hand, the "a trous" option might be negatively affecting the selection of features in the surroundings on the potholes. Also, we suspect that some over-fitting is also happening due to a larger gap between validation and testing performance compared to the other Faster R-CNN approaches.

TABLE III
AVERAGE INFERENCE TIME FOR THE 3 POTHOLE DETECTORS [3] AND DIFFERENT GPGPU DEVICES.

	Inference time (ms)	
Models	**Tesla P100M 16GB**	**DrivePX2 TegraA**
inception_v2	53.2	177.1
resnet101	94.2	432.7
inception_resnet_v2_atrous	172.1	732.9

Finally, *Faster R-CNN Inception v2* yields modest values of AP at lower processing cost, which is also a valid pothole detector for the aim of Highway Pilot use case within AUTOPILOT. Rates up to 6fps when running on NVidia DrivePX2 could be achieved.

Figures 7 to 9 depict some samples of the pothole detections with the best performing model, which is **Faster R-CNN Resnet101**. The potholes were correctly found in the scenes without any road surface filtering or manual pre-selection of image region of interest. The captions on the images describe the details of every case. We provide a supplementary video to demonstrate the achieved detection performance on a real world scene.

VI. CONCLUSION

This paper has presented how to automatically catch road potholes in worldwide real scenes. We built an image dataset with high intraclass variance from several sources with the aim of learning robust and general pothole appearance models that can aid the detection of this type of road hazards. We fine-tuned Deep Neural Networks for object detection with demonstrated good performance in the state of the art. We validated the suitability of some of these models for the particular goal of road pothole detection. We concluded that precise localization of potholes is challenging due to ground-truth errors during annotation because of the nature of road potholes. Thus, we relaxed the Jaccard index to $IoU = 0.4$ for the evaluation. Consequently, *Faster R-CNN Resnet101* achieved averaged precision values of 82%, while *Faster R-CNN Inception v2* yielded 75% at a lower processing cost. The latter one, when tested on Nvidia DrivePX2 Autochauffeur platform, it could run at 5-6fps. Furthermore, the pothole detector was deployed in a real vehicle as part of AUTOPILOT project.

One of the limitations of the detector is the runtime in case of real-time requirements on the target system. For AUTOPILOT, the current frame rate is valid because potholes

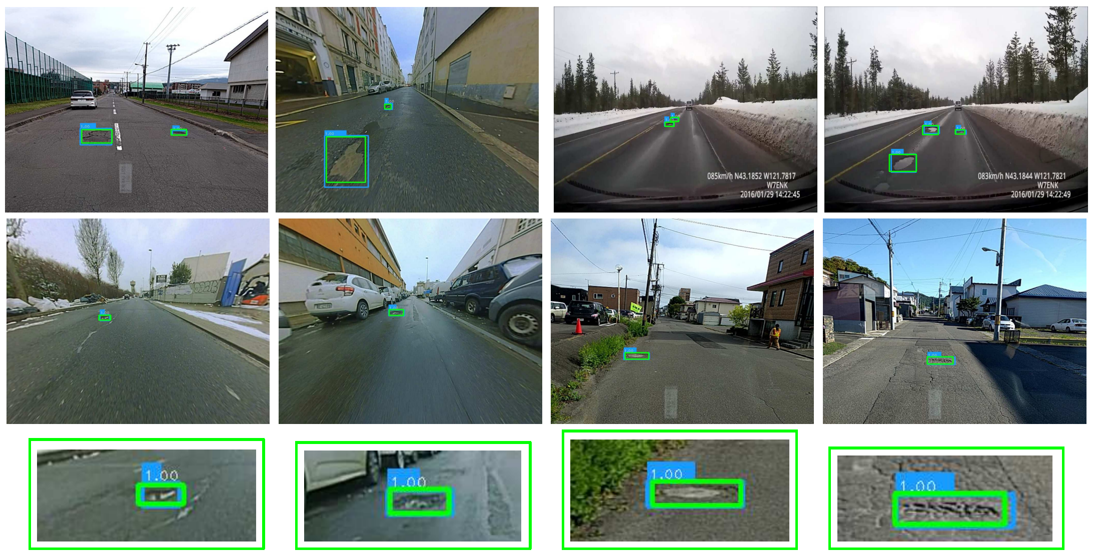
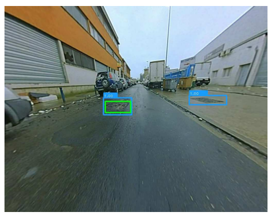
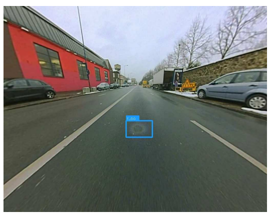
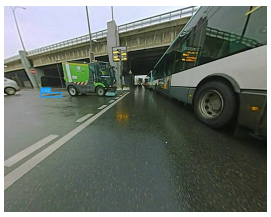
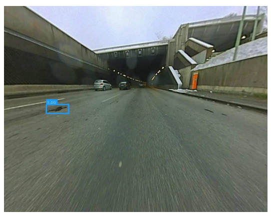


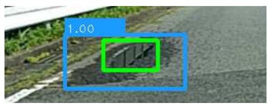
Fig. 7. Correct pothole detections in the test set using the model *Faster R-CNN Resnet101*. The ground-truth boxes are in green colour and the detections with overlaid score are in blue. The first row shows multiple detections, the second row are single potholes that are zoomed in and displayed in the third row.


(a) Patch/pothole not annotated

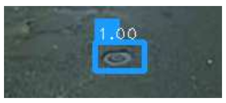

(b) Oil smear similar to pothole


(c) Water puddle at far distance

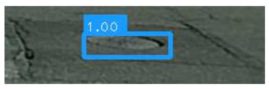

(d) Stain on road surface



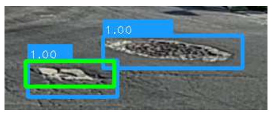
(e) Correct detection but bigger than ground truth due to wet surface



(f) Small manhole within shallow road patch

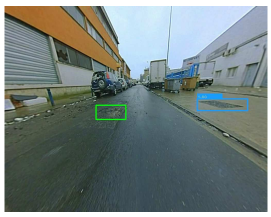


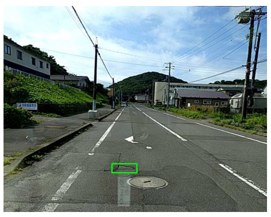
(g) Manhole with shallow border

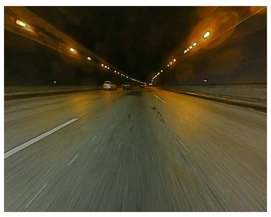


(h) Manhole within shallow patch

Fig. 8. Examples of false positive detections in the test images using the model *Faster R-CNN Resnet101*.


(a) FN in green, but correctly detected in adjacent frames of the sequence


(b) FN in green and TN manhole both not detected as potholes


(c) Surface stains inside tunnel and not detected as expected

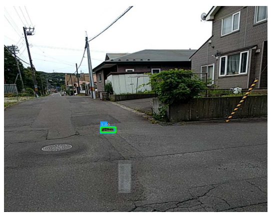

(d) Correct behaviour not detecting manhole

Fig. 9. Examples of true and false negatives in the test images using the model *Faster R-CNN Resnet101*.

will be notified to IoT platforms upon detection and they will later serve road hazard warnings, hence, immediate reactive vehicle manoeuvres are not targeted. In addition, we have observed that most of the false positives are due to manholes. Sometimes, they are in a shallow road patch which could be considered as correct detection. Other false detections outside road boundaries could be easily filtered by using lane/road detectors. Moreover, in highly damaged road surfaces the detector found unlabelled potholes and some road cracks, which could be considered as correct due to the bad condition of the pavement. Also, a limitation of the algorithm is high speed driving (>60Km/h). Images show blur that filters edges and fine-grained details making very difficult the detection task. This limitation comes from the acquisition system, camera settings, illumination conditions and other related factors.

For the future we plan DNN optimizations to accelerate inference. One possibility is to split specific parts of the neural network in to CPU or GPU processing. Also, the evaluation of other lightweight networks that can be trained from scratch on the pothole dataset or the reduction of layers in state-of-the-art networks. For instance, YOLOv3 [48] has recently reported state-of-the-art accuracy at faster inference time. For fine-tunning during learning, enlarging the dataset can help improve the detection ratios. Additionally, other road hazards of interest will be studied in the future as part of the AUTOPILOT project.

Acknowledgement

This work has received funding from the European Research Council (ERC) under the European Union's H2020 research and innovation programme (grant agreement no. 731993, project AUTOPILOT). The authors would like to thank Valeo Comfort and Driving Assistance (VCDA) SAS for the provision of a equipped vehicle and video recordings as part of AUTOPILOT. Also, the authors would like to thank Dr. M. Nieto and Dr. S. Sanchez that helped with the proofreading.

References

[1] H. Maeda, Y. Sekimoto, T. Seto, T. Kashiyama, and H. Omata, "Road Damage Detection Using Deep Neural Networks with Images Captured Through a Smartphone," *ArXiv e-prints*, Jan 2018. [Online]. Available: http://arxiv.org/abs/1801.09454

[2] R. Madli, S. Hebbar, P. Pattar, and V. Golla, "Automatic detection and notification of potholes and humps on roads to aid drivers," *IEEE Sensors Journal*, vol. 15, no. 8, pp. 4313–4318, Aug 2015.

[3] S. Nienaber, M. Booysen, and R. Kroon, "Detecting Potholes Using Simple Image Processing Techniques and Real-World Footage," in *34th Annual Southern African Transport Conf. (SATC)*, 2015, pp. 153–164.

[4] "Global Transport Scenarios 2050," Last viewed: 2018-06-14. [Online]. Available: https://www.worldenergy.org/wp-content/uploads/2012/09/wec_transport_scenarios_2050.pdf

[5] "Ministerio de fomento," Last viewed: 2018-05-24. [Online]. Available: http://www.fomento.gob.es

[6] M. Gavilán, D. Balcones, O. Marcos, D. F. Llorca, M. A. Sotelo, I. Parra, M. Ocaña, P. Aliseda, P. Yarza, and A. Amírola, "Adaptive road crack detection system by pavement classification," *Sensors*, vol. 11, no. 10, pp. 9628–9657, Oct 2011.

[7] J. Janai, F. Güney, A. Behl, and A. Geiger, "Computer Vision for Autonomous Vehicles: Problems, Datasets and State-of-the-Art," *ArXiv e-prints*, apr 2017. [Online]. Available: http://arxiv.org/abs/1704.05519

[8] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Wiley, 2010.

[9] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[10] "Durable Pothole Repairs," Last viewed: 2018-06-15. [Online]. Available: https://trimis.ec.europa.eu/project/durable-pothole-repairs

[11] "AUTOPILOT H2020 project," Last viewed: 2018-05-24. [Online]. Available: http://autopilot-project.eu/

[12] U. Bhatt, S. Mani, E. Xi, and J. Z. Kolter, "Intelligent Pothole Detection and Road Condition Assessment," *ArXiv e-prints*, Oct 2017. [Online]. Available: http://arxiv.org/abs/1710.02595

[13] S. Lee, S. Kim, K. E. An, S.-K. Ryu, and D. Seo, "Image Processing-based Pothole Detecting System for Driving Environment," in *IEEE Intl. Conf. on Consumer Electronics (ICCE)*, Jan 2018, pp. 0–1.

[14] S. Hegde, H. Mekali, and G. Varaprasad, "Pothole detection and inter vehicular communication," in *IEEE Intl. Conf. on Vehicular Electronics and Safety (ICVES)*, Dec 2014, pp. 84–87.

[15] A. Mednis, G. Strazdins, R. Zviedris, G. Kanonirs, and L. Selavo, "Real time pothole detection using Android smartphones with accelerometers," in *IEEE Intl Conf. on Distributed Computing in Sensor Systems and Workshops, (DCOSS)*, Jun 2011, pp. 1–6.

[16] H.-W. Wang, C.-H. Chen, D.-Y. Cheng, C.-H. Lin, and C.-C. Lo, "A Real-Time Pothole Detection Approach for Intelligent Transportation System," *Mathematical Problems in Engineering*, vol. 2015, pp. 1–7, Aug 2015.

[17] J. Ren and D. Liu, "PADS: A reliable pothole detection system using machine learning," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer, Dec 2017, pp. 327–338.

[18] A. Fox, B. V. Kumar, J. Chen, and F. Bai, "Multi-Lane Pothole Detection from Crowdsourced Undersampled Vehicle Sensor Data," *IEEE Transactions on Mobile Computing*, vol. 16, no. 12, pp. 3417–3430, Dec 2017.

[19] J. Almazán, L. M. Bergasa, J. J. Yebes, R. Barea, and R. Arroyo, "Full auto-calibration of a smartphone on board a vehicle using IMU and GPS embedded sensors," in *IEEE Intelligent Vehicles Symposium (IV)*, Jun 2013, pp. 1374–1380.

[20] C. M. Jengo, S. C. Engineer, A. Way, and I. Curtis, "Pothole Detection and Road Condition Assessment Using Hyperspectral Imagery," *Proceedings of the American Society for Photogrammetry & Remote Sensing (ASPRS)*, pp. 7–11, 2005.

[21] T. Kim and S.-k. Ryu, "System and Method for Detecting Potholes based on Video Data 1 1," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 5, no. 9, pp. 703–709, 2014.

[22] A. Akagic, E. Buza, and S. Omanovic, "Pothole Detection: An Efficient Vision Based Method Using RGB Color Space Image Segmentation," in *Intl. Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, May 2017, pp. 1104–1109.

[23] S. Nienaber, R. S. Kroon, and M. J. Booysen, "A comparison of low-cost monocular vision techniques for pothole distance estimation," in *IEEE Symposium Series on Computational Intelligence, (SSCI)*, Dec 2015, pp. 419–426.

[24] D.-W. Jang and R.-H. Park, "Pothole detection using spatio-temporal saliency," *IET Intelligent Transport Systems*, vol. 10, no. 9, pp. 605–612, Nov 2016.

[25] K. Azhar, F. Murtaza, M. H. Yousaf, and H. A. Habib, "Computer vision based detection and localization of potholes in asphalt pavement images," in *IEEE Canadian Conf. on Electrical and Computer Engineering (CCECE)*, May 2016, pp. 1–5.

[26] K. E. An, S. W. Lee, S.-K. Ryu, and D. Seo, "Detecting a pothole using deep convolutional neural network models for an adaptive shock observing in a vehicle driving," in *IEEE Intl. Conf. on Consumer Electronics (ICCE)*, jan 2018, pp. 1–2.

[27] I. Moazzam, K. Kamal, S. Mathavan, S. Usman, and M. Rahman, "Metrology and visualization of potholes using the microsoft kinect sensor," in *IEEE Intelligent Transportations Systems Conf. (ITSC)*, Oct 2013, pp. 1284–1291.

[28] Z. Zhang, X. Ai, C. K. Chan, and N. Dahnoun, "An efficient algorithm for pothole detection using stereo vision," in *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 564–568.

[29] A. Mikhailiuk and N. Dahnoun, "Real-time pothole detection on TMS320C6678 DSP," in *IEEE Intl. Conf. on Imaging Systems and Techniques (IST)*, Oct 2016, pp. 123–128.

[30] B. H. Kang and S. I. Choi, "Pothole detection system using 2D LiDAR and camera," in *IEEE Intl. Conf. on Ubiquitous and Future Networks (ICUFN)*, Jul 2017, pp. 744–746.

[31] N. J. Sucgang, M. R. Jr, and N. A. Arriola, "Road Surface Obstacle Detection using Vision and LIDAR for Autonomous Vehicle," in *Intl. MultiConf. of Engineers and Computer Scientists (IMECS)*, vol. I, March 2017, pp. 3–7.
[32] "Pothole Detection and Warning System - Jaguar Land Rover," Last viewed: 2018-05-14. [Online]. Available: https://www.landrover.com/experiences/news/pothole-detection.html
[33] Google patent, "System That Detects Potholes," Last Viewed: 2018-05-14. [Online]. Available: https://www.digitaltrends.com/android/google-pothole-detection/
[34] Y. Jo and S. Ryu, "Pothole detection system using a black-box camera," *Sensors*, vol. 15, no. 11, pp. 29 316–29 331, 2015.
[35] P. F. Felzenszwalb, R. B. Girshick, D. Mcallester, and D. Ramanan, "Object Detection with Discriminatively Trained Part Based Models," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 32, no. 9, pp. 1–20, Sep 2009.
[36] J. J. Yebes, L. M. Bergasa, and M. GarcÃa-Garrido, "Visual object recognition with 3d-aware features in kitti urban scenes," *Sensors*, vol. 15, no. 4, pp. 9228–9250, 2015.
[37] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017, pp. 3296–3297.
[38] "Tensorflow detection model zoo," Last viewed: 2018-05-14. [Online]. Available: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md
[39] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun 2017.
[40] R. Girshick, "Fast R-CNN," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Dec 2015, pp. 1440–1448.
[41] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *ArXiv e-prints*, Apr. 2017. [Online]. Available: https://arxiv.org/abs/1704.04861
[42] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," feb 2016. [Online]. Available: http://arxiv.org/abs/1602.07261
[43] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Eur. Conf. on Computer Vision (ECCV)*. Springer International Publishing, 2016, pp. 21–37.
[44] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation," *Computing Research Repository*, vol. abs/1801.04381, 2018. [Online]. Available: http://arxiv.org/abs/1801.04381
[45] "Common Objects in COntext - COCO," Last viewed: 2018-12-01. [Online]. Available: http://cocodataset.org
[46] Tzutalin, "{LabelImg} is a graphical image annotation tool that labels object bounding boxes in images," 2015. [Online]. Available: https://github.com/tzutalin/labelImg
[47] M. Abadi, A. Agarwal, P. Barham, and e. a. Brevdo, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," *ArXiv e-prints*, 2016. [Online]. Available: http://arxiv.org/abs/1603.04467
[48] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *Computing Research Repository*, vol. abs/1804.02767, 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

J. Javier Yebes received the Ph.D. degree (Cum Laude) in the field of Computer Vision and Robotics from the University of Alcalá (UAH), Spain, in 2014. He is a Researcher in computer vision and machine learning at Vicomtech, in the Department of ITS and Engineering, where he has a technical leading role for European H2020 projects. Also, he was the technical leader of an R&D project for railways industry at Gobotix Ltd. (UK) in 2015-2017. He was pre- and post-Doctoral assistant at RobeSafe research group, UAH, from 2009 to 2015. His research interests include computer vision, deep learning, autonomous vehicles, ADAS, IoT and cloud.

David Montero received the M.Sc. in Industrial Engineering in 2017 and the B.Sc. in 2014 from the University of Seville, Spain. He is a Researcher at Vicomtech, in the Department of ITS and Engineering where he is working in several private and public funded projects. During his studies, he worked on SLAM, 3D reconstruction and obstacle avoidance for UAVs. His research interests include computer vision and deep learning applied to autonomous vehicles and drones.

Ignacio Arriola received the M.Sc. in Computational Engineering and Intelligent Systems in 2018 and the B.Sc. in Electrical Engineering in 2017 from the University of Pais Vasco (UPV/EHU), Spain. During his degree thesis, he investigated the identification of road type based on driving patterns. He is currently an Assistant Researcher on computer vision and deep learning at Vicomtech. His research interests are in those fields applied to autonomous vehicles and ITS.

4.3 Multi-Camera BEV Video-Surveillance System for Efficient Monitoring of Social Distancing

- **Authors:** David Montero and Nerea Aranjuelo and Peter Leskovsky and Marcos Nieto and Naiara Aginako
- **Journal:** Multimedia Tools and Applications
- **Volume:** -
- **Pages:** -
- **Year:** 2023 (pending publication)
- **Publisher:** Springer

Multi-Camera BEV Video-Surveillance System for Efficient Monitoring of Social Distancing

David Montero[1*], Nerea Aranjuelo[2,1], Peter Leskovsky[2], Estíbaliz Loyo[2], Marcos Nieto[2] and Naiara Aginako[1]

[1*]Computer Vision and Artificial Inteligence, University of the Basque Country, Donostia, 20018, Guipuzcoa, Spain.
[2]ITS and Engineering, Vicomtech, Donostia, 20009, Guipuzcoa, Spain.

*Corresponding author(s). E-mail(s): davidsfc10@gmail.com;
Contributing authors: naranjuelo@vicomtech.org; pleskovsky@vicomtech.org; eloyo@vicomtech.org; mnieto@vicomtech.org; naiara.aginako@ehu.eus;

Abstract

The current sanitary emergency situation caused by COVID-19 has increased the interest in controlling the flow of people in indoor infrastructures, to ensure compliance with the established security measures. Top view camera-based solutions have proven to be an effective and non-invasive approach to accomplish this task. Nevertheless, current solutions suffer from scalability problems: they cover limited range areas to avoid dealing with occlusions and only work with single camera scenarios. To overcome these problems, we present an efficient and scalable people flow monitoring system that relies on three main pillars: an optimized top view human detection neural network based on YOLO-V4, capable of working with data from cameras at different heights; a multi-camera 3D detection projection and fusion procedure, which uses the camera calibration parameters for an accurate real-world positioning; and a tracking algorithm which jointly processes the 3D detections coming from all the cameras, allowing the traceability of individuals across the entire infrastructure. The conducted experiments show that the proposed system generates robust performance indicators and that it is suitable for real-time applications to control sanitary measures in large infrastructures. Furthermore, the proposed projection approach achieves an average positioning error below 0.2 meters, with an improvement of more than 4 times compared to other methods.

Keywords: Crowd monitoring, Multi-Camera tracking, 2D-3D projection, COVID-19

1 Introduction

Since the rise of COVID-19 in December 2019, numerous studies have emerged to help stop the disease spreading, tackling the problem from different perspectives [1, 2]. Most of these studies focus on preventing the spread of the virus by proposing health and social measures, and methods to ensure its compliance. Among these prevention measures, like improving the ventilation in indoor areas [3] and using medical masks [4, 5], one key measure that has been adopted by all the governments is social distancing. Recent research has confirmed the evidence that maintaining a

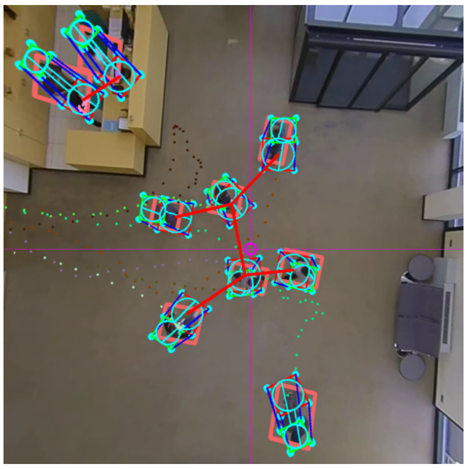


Fig. 1 Visual example of the proposed system operation. For each 2D detection, the best fitting 3D cylinder is estimated in real-world coordinates. After a 3D tracking step, the desired metrics are computed.

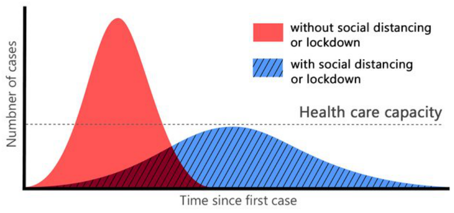


Fig. 2 Gaussian distribution of infection transmission rate for a given population, with and without social distancing obligation. This figure was originally presented in [8]

social distance of 1.6 to 2 meters highly reduces the disease spreading [6, 7], as shown in Fig. 2.

Thus, an important need has arisen to create applications that are capable of monitoring people in indoor and outdoor infrastructures to guarantee a safe interpersonal distance, respect the indoor capacity limitation, know the most crowded time intervals or track subjects who violate the established measures.

A full variety of solutions have been proposed, addressing the problem from different perspectives, such as using wi-fi signals [9], wearable devices [10], drones [11], mobile robots [12], etc.

Among all these methodologies, camera-based solutions have proven to be an effective, non-invasive and affordable alternative to accomplish this task. Furthermore, these methods can take advantage of the camera infrastructure already available in smart cities, facilitating their scalability and sustainability. Within the camera-based solutions, we find different approaches, such as face recognition [13–15] or crowd density estimation [16–18]. However, for the considered use case, the approach that works best is the one that combines human detection and tracking [8, 19–23], since it allows covering wider areas than with face recognition, and makes it possible to collect spatio-temporal information about the individuals who are violating the measures (unlike solutions based on density estimation). Nevertheless, most of the systems based on this methodology only work with side or frontal view perspectives. In this setup, cameras still produce an important amount of occlusions, especially when dealing with very large and crowded surfaces (e.g. airports or shopping centers). These occlusions reduce the accuracy of the tracking algorithms and of the 3D detection projection, which are crucial for computing COVID-19-related performance indicators (PIs), such as the interpersonal distance or the indoor capacity limitation.

These problems can be mitigated by using algorithms that detect people on the Bird's-Eye-View (BEV) domain and omnidirectional cameras. Thus, the number of occlusions are minimized and the covered area is maximized, as the cameras are placed in the ceiling. Therefore, the top-view perspective makes this approach suitable for applications related with the compliment of the COVID-19 sanitary measures. Although there is some recent research [24, 25], this topic still remains underexplored, as the proposed solutions only work with single-camera scenarios and for very limited monitoring areas.

For this reason, we present a multi-camera BEV people flow monitoring system, capable of extracting reliable real-time PIs in extremely large infrastructures, such as airports or shopping centers. The proposed system relies on three main pillars: an optimized top view human detection neural network based on YOLO-V4, capable of working with data from cameras at different heights; a multi-camera 3D detection projection and fusion procedure, which uses the camera calibration parameters for an accurate real-world positioning; and a tracking algorithm which jointly processes the 3D detections coming from all the cameras.

We highlight two novel contributions in this work:

- A modification in the traditional pipeline that allows our system to operate efficiently in multi-camera environments. Unlike the rest of the proposed methods, we move the projection step to real world coordinates just after the detection step for each camera (instead of applying it after the tracking step). Then, thanks to an initial multi-camera calibration procedure, we are able to track the subjects uninterruptedly all over the monitored area using just a single tracker instance for processing all the detections. Furthermore, this approach allows using cameras installed at different heights, since it does not take into account the detection bounding box for the multi-camera fusion but the real position in meters.
- A 3D projection and multi-camera fusion procedure. Using the intrinsic and extrinsic parameters of the involved cameras, it estimates the best fitting 3D cylinder for each detected bounding box and fuses the cylinders of the overlapping regions of the camera views that belong to the same person. This corrects possible occlusion problems and allows us to expand the useful range of the cameras.

We conduct different experiments to demonstrate that the proposed system generates robust PIs and that it is suitable for real-time applications to control sanitary measures, such as guaranteeing a safe interpersonal distance, respecting the indoor capacity limitations, identifying the most crowded time intervals or tracking subjects who violate the established measures. Furthermore, the proposed projection approach achieves an average positioning error below 0.2 meters, with an improvement of more than 4 times compared to other methods. An example of the proposed system operation is shown in Fig. 1.

The rest of the paper is organized as follows. First, we present a review of the related work in Section 2. Section 3 describes the proposed method. In Section 4 we provide experimental results. A discussion about the method and the results is presented in Section 5 Finally, conclusions are given in Section 6.

2 Related Work

2.1 Social Distance Monitoring

In recent months, several methods aiming to monitor compliance with sanitary measures have been proposed, specially for social distance monitoring [26, 27], addressing the problem from different perspectives. For example, in [12] the authors developed a mobile robot for social distance monitoring in crowded scenarios. It was equipped with an RGB-D camera and a 2D lidar to make collision-free navigation in mass gatherings. They used YOLO-V3 Deep Neural Network (DNN) [28] along with Deep SORT algorithm [29] for detection and tracking of individuals, respectively. However, the limited field of view of the robot and the cost of acquiring and maintaining several robots make this solution unsuitable for large infrastructures. In [11], the authors use a drone to deploy a social distance monitoring system. The drone detect human heads in realtime and then calculate the social distancing between pedestrians on UAV images using a DNN that follows the PeleeNet as backbone and further incorporates the multi-scale features and spatial attention to enhance the features of small objects. In [30], another drone-based method for social distance monitoring was proposed. Relying on the drone's camera and YOLO-V3 algorithm, the system was able to detect people from side or frontal-view images and to monitor if the social distance was respected and if subjects were wearing masks. Nevertheless, these drone-based solutions are only valid for outdoor environments and they have a high associated cost due to the drone acquisition and maintenance. On the other hand, in [10] the authors used a wearable, oscillating magnetic field-based proximity device for social distance monitoring. Despite this solution achieves excellent results in indoor and outdoor environments, it is unfeasible as it requires all subjects to wear the device.

Among all these innovative solutions, systems using vision-based human detection have proven to be the best value for money, as they only need a monocular camera and a GPU-enabled server for real-time people monitoring and they can cover wide areas. Furthermore, they are also less intrusive than other methods mentioned before. Rezaei and Azarmi [8] proposed a pedestrian-detection-based social distance monitoring system. Using the YOLO-V4 [31] model pretrained with COCO dataset and SORT tracking algorithm [32] they were able to operate accurately in real-time. In [21] and [22], the authors also proposed a social distance monitoring system based on YOLO-V3 DNN with Deep SORT tracking algorithm and YOLO-V4 respectively. Su et al. [19] follow the same pipeline and combine the euclidean distance with spatio-temporal information about the trajectory of the pedestrians to better understand the scene. Shorfuzzaman et al. [20] propose to add a perspective transformation to bird-eye-view to determine the ROI in which social distancing will be monitored, but they do not add a tracking step. However, all these systems suffer from the same problem. Pedestrian occlusions are very common when dealing with large or crowded scenarios and using frontal or side-view images.

This problem can be mitigated by using a BEV perspective with omnidirectional cameras. Thus, the occlusions are minimized in the central area of the camera and the covered area is maximized. This approach was adopted in [24] and [25]. Nevertheless, their proposed systems works only with the central area of a single camera, where the occlusions need not to be considered. Therefore, for covering wide areas, an important number of cameras would be needed. This increases the hardware requirements and the cost of the system, and makes it unsuitable for large infrastructures. Furthermore, the question of joining tracks across camera views would have to be addressed too.

To overcome these problems, our proposed multi-camera BEV people flow monitoring system uses a multi-camera detection fusion procedure. The 2D detections received from the detection DNN are projected to real-world coordinates and the best-fitting 3D cylinder is estimated for every given detection. Then, the detections of the overlapped cameras are fused, correcting possible occlusion problems and allowing us to expand the useful range of the cameras. Finally, the 3D trajectories are effectively computed by our online 3D version of the tracking algorithm proposed in [33]. This way, we only need to use a single tracker for all the cameras to track the subjects over the entire monitored area, no matter in which camera's view is detected.

2.2 Overhead Human Detection-Based Tracking

In recent years, several tracking algorithms have been proposed to deal with overhead people detections. Ahmed and Adnan [34], proposed rHOG, an overhead tracking algorithm which uses the variable size bounding boxes with different orientations, with respect to the radial distance of the center of the image. In [35], the authors proposed a people tracking algorithm for industrial environments that works with motion blobs gathered by an overhead camera. This algorithm, based on rHOG, uses the history of already imaged population with the anticipated blob position of the person observed. Other works base their algorithms on Kalman [36] or particle filters [37].

Although these algorithms work well with a single camera, they are not suitable for multi-camera scenarios, as they are not able to merge data coming from multiple overlapped cameras. We could add a fusion stage after the tracking process, but that would require having multiple tracking instances calibrated for each camera, which is inefficient and complicates the system installation process.

For this reason, we propose to alter the order of the stages: first, to project all the detections in the real world and combine the ones in the overlapping areas and, then, to apply a single 3D tracking process. Thus, we adapt the 3D offline algorithm presented in [33] for the considered use case, resulting in an online version optimized to work with human detections described in Section 3.7.

3 Proposed Method

3.1 Problem Definition

We consider the problem of efficient monitoring of a set of established security measures in large infrastructures using non-invasive technology. More specifically, we aim to create a video-surveillance system capable of monitoring compliance with social distance and capacity limitation measures, as well as tracking the offenders or the possibly infected subjects. Therefore, the system must be capable of merging the information coming from multiple cameras to track subjects all over the monitored region. We use

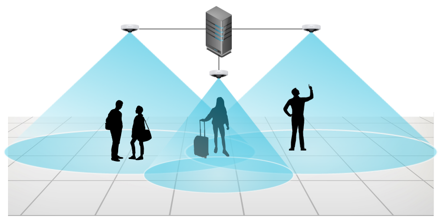


Fig. 3 Illustration of the considered scenario. Multiple omnidirectional cameras with a small overlap cover the monitored area. The cameras are connected to a central GPU-enable server.

overlapped camera views to track people across views.

For this task, the system will extract the necessary information from a set of cameras placed on the ceiling of the monitored infrastructure. The number of cameras depends on the area to be covered and on the height of the ceiling. The set up needs to guarantee that all the areas of interest are visible by the cameras with a minimum resolution (limited by the capabilities of the human detection network). The omnidirectional cameras with fish-eye lenses have a wide field of view, which makes them appropriate for monitoring large areas with a minimum number of sensors. Intrinsic and extrinsic parameters of all the cameras need to be available for an accurate distance measuring. An image illustrating the considered use case is presented in Fig. 3. The system must be able to report reliable and real-time information about the state of measures compliance using minimum processing requirements, preferably a single-GPU server.

3.2 System Overview

An overview graphical diagram of the proposed workflow is shown in Fig. 4 and a flowchart in Fig. 5.

The system requires an initial camera calibration process (Section 3.3). First, the images are grabbed from each configured camera. Then, they are preprocessed in parallel using CUDA library and fed into the pruned version of YOLO-V4 detector, implemented in TensorRT framework (Section 3.4). Therefore, from the original (distorted) images, a set of detections is obtained for each camera c at each frame t, $\mathcal{D}_c = \{D_{c,t,i}\}$. Each detection is modeled as a 4-point rectangle in image coordinates $D = (x_0, y_0, x_1, y_1, x_2, y_2, x_3, y_3)$.

As discussed in Section 2.2, we alter the typical order of the tracking and projection stages for two reasons: it is easier to estimate the trajectory using world coordinates than image coordinates from an omnidirectional camera (where the bounding box varies rapidly); and we only need a single tracker instance to process all detections in world coordinates. Thus, from detections $\mathcal{D}_c$ a fitting process yields the desired 3D cylinder shapes $\mathcal{C}_c = \{C_{c,t,i}\}$, where each cylinder is encoded as $C = (X, Y, H, r)$, where (X, Y) is the center point in the XY plane and (H, r) represent its height and radius, respectively. Fitting process is explained in Section 3.5. Then, a fusion mechanism determines which cylinders correspond to the same object for cameras with overlapped fields of view. Let's denote fused cylinders as C, where we have removed the c sub-index, as now all cylinders are expressed in global 3D coordinates and not related anymore to any specific camera. Fusion procedure is presented in section 3.6. Next, the tracking stage takes these cylinders, applying a constant-velocity predicting model, plus managing appearance and disappearance of objects, miss-detections, etc. The tracking mechanism is explained in section 3.7. Note that tracks are expressed as follows: $\mathcal{T} = \{T_k\}$, where $T = (X, Y, H, r, \dot{X}, \dot{Y}, \ddot{X}, \ddot{Y})$ to account for the derivative dimensions for prediction phase. Finally, the output time-consistent tracks are analysed to extract the necessary information for the monitoring of the established security measures (Section 3.8).

3.3 System Calibration

When the system is set up, and prior to the first operation, a calibration step is needed in order to compute the intrinsic and extrinsic parameters. The estimated camera parameters are used for mapping the image coordinates to the 3D world coordinates and for locating each camera with respect to the others. Consequently, we are able not only to get the real position of each person detected in the scene but also to detect when they move from one camera to another and to merge the detections from overlapped cameras.

The well-known fish-eye camera model is used in this project, where the projection process is

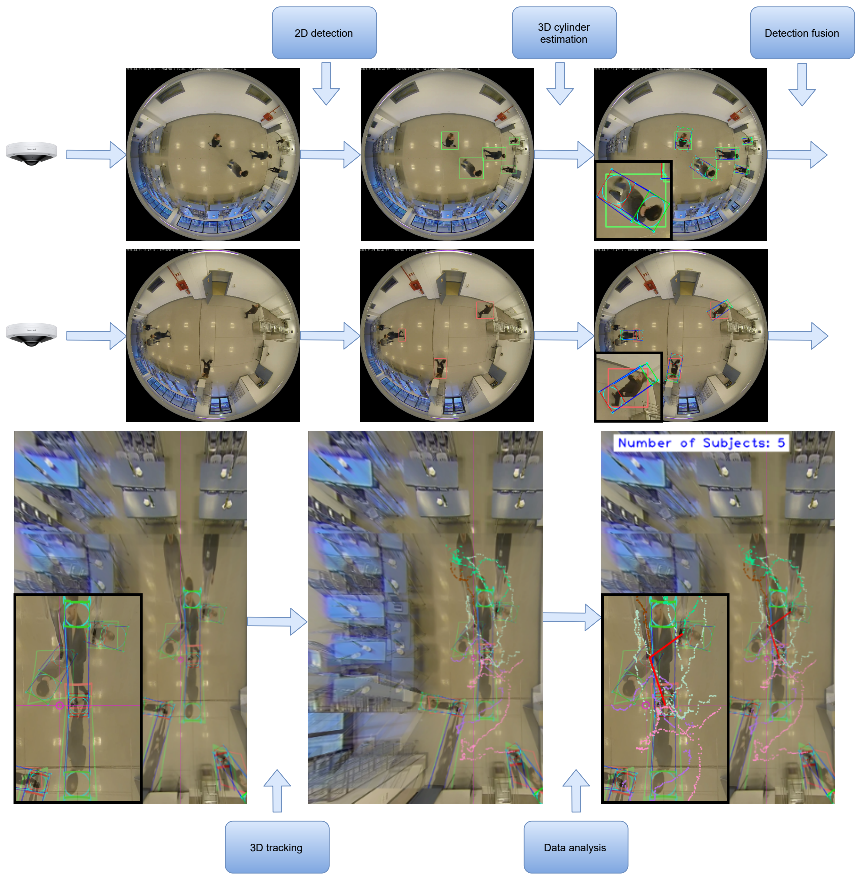


Fig. 4 Overview graphical diagram of the proposed workflow. Note that the rectified images (in the lower row) are generated only for visualization purposes and are not necessary for fusing the detections, tracking or data analysis.

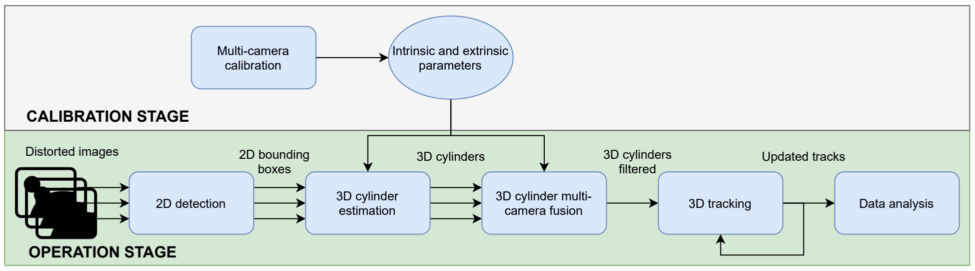


Fig. 5 Overview flowchart of the proposed workflow. Note that until the cylinders fusion step, the detections of each camera are processed separately.

governed by a fish-eye distortion vector $\mathbf{k} = (k_1, k_2, k_3, k_4)$ and a linear projection matrix K which holds the focal length and principal point parameters. As a result of the calibration, for each camera, the intrinsic (K, and $\mathbf{k}$) and extrinsic (rotation matrix R and translation $\mathbf{t}$) are obtained. The pose of each camera is expressed with respect a common 3D point used as world reference.

Any 3D point in the world $\mathbf{X} = (X, Y, Z, 1)^\top$ expressed in homogeneous coordinates can then be projected into any of the images. First by representing the point with respect to the camera coordinate system, $\mathbf{X}_c = P_c\mathbf{X}$, where $P_c = (R|\mathbf{t};\ \mathbf{0}|1)$ is the 4×4 corresponding pose matrix. The fish-eye distortion model is then applied on the 3D rays joining $\mathbf{X}_c$ and the camera optical center, by defining $a = X_c/Z_c$ and $b = Y_c/Z_c$, and $r^2 = a^2 + b^2$. The longitude angle $\theta = arctan(r)$. The distortion vector $\mathbf{k}$ is then applied to obtain the angle of incidence $\theta_d = \theta(1 + k_1\theta^2 + k_3\theta^4 + k_3\theta^6 + k_4\theta^8)$. The point in the normalized domain is then obtained as $(x', y') = (a\theta_d/r, b\theta_d/r)$, and its projection into the image domain as $(u, v, w) = K(x', y', 1)^\top$ (pixel values are obtained as $x = u/w$ and $y = v/w$).

The calibration can be used as well to re-project any point in the image plane (x, y) into a 3D ray starting from the optical center of the camera, applying $\mathbf{r} = (u', v', w') = K^{-1}(x, y, 1)^\top$ and normalizing so $r = 1$. Then, if a 3D world plane is selected (e.g. $Z = 0$), the intersection of the 3D ray with the plane determines a 3D point in the world. This is useful to re-project 2D image points of objects in the ground plane to obtain their position in the XY plane in world coordinates (note this assumption holds true only if the 2D image point correspond to an object or part of object which is touching or at the ground level).

Retrieving the 3D ray $\mathbf{r}$ from pixels (x, y) implies inverting the fish-eye distortion vector, which can be accomplished using iterative minimization processes (we are using OpenCV's implementation). In addition, to speed up the re-projection process, it is recommended to create remap functions by pre-computing the mapping relation between points in the images and the 3D space, giving as a result the ability to create rectified and Bird's-eye View (BEV) of multiple cameras in a single step.

3.4 People Detection using Overhead Cameras

Similar to [38], we train YOLO-V4 object detection Convolutional Neural Network (CNN) to detect people directly in overhead images from fish-eye cameras. We use this single-stage detector because it provides a good balance between accuracy and inference time. In a multi-camera system it is important to guarantee a fast inference for a real-time analysis. Compared to previous versions, YOLO-V4 includes detections at three scales, which improves the small object detection accuracy.

Our aim is to design a system capable of working with a camera installed 3 to 10 meters high, so that it is suitable for different large space scenarios. Consequently, our detector should work on this height range. Even if the YOLO-V4 model provides detections at different scales, the objects' scale varies considerably for such a big range. In order to ensure the robustness of the model no matter the height of the camera, we add an image scaling step previous to the detection, which resizes the image to guarantee that the people size in the center of the image is stable no matter the installation height (approximately 20×20 pixels). In addition, we train two models, one targeted for the lowest heights (3-6 m) and another for the highest installations (6-10 m).

As there is no public dataset with top-view fish-eye images of large spaces focused on human detection and multi-camera systems, we use several recordings to build our training dataset. We set up two omnidirectional cameras installed at 3.3 meters and another camera at 8 meters. We capture 10,000 images for the lower height range and 10,000 images for the upper one. In addition, to augment both ranges' data variety we add 5,600 synthetic images from the Advanced Synthetic Dataset presented in [38] to each of the datasets. We manually annotate the captured data. As shown in [39], rotation and histogram equalization are some of the most efficient image augmentations for training accurate object detection CNNs. Consequently, we apply rotations, flipping and histogram equalization augmentations (CLAHE) to our images. We randomly combine these augmentations and generate 4 new samples for each image. The images are resized to 512×512 for the models training.

We train both models on a NVIDIA Tesla P100 using Darknet framework [31]. We initialize the models with pre-trained weights on the MS COCO dataset [40] and train them for $40,000$ iterations with a learning rate of $0,001$ and a weight decay of $0,0005$. We use the stochastic gradient descent optimizer with a batch size of 64 images.

To further increase the performance of the model we apply two optimization processes. First, we apply the weight pruning procedure described in [41]. It is an iterative process with three stages in each iteration:

- Network training penalizing the scaling weights of the batch normalization layers in the cost function.
- Network pruning percentage of convolutional filters corresponding to the lowest batch normalization scaling weights.
- Pruned network fine-tuning without penalization.

This procedure is repeated until the desired balance between precision and speed is reached. In our case, we pruned each network 3 times eliminating 50%, 50% and 70% of the remaining filters, keeping at least 10% of filters in each layer.

Finally, the pruned models are ported to TensorRT framework to apply hardware-level optimizations. With these optimizations we are able to reduce the inference time almost a 90% for each model, from 22 fps to 110 fps.

3.5 3D Cylinder Estimation

The next stage consists of transferring detections $\mathcal{D}_c$ from the 2D domain of all camera images to the 3D real world domain using the intrinsic and extrinsic camera parameters. More specifically, for each 2D detection $D_{c,t,i}$ we estimate the 3D cylinder $C_{c,t,i}$ whose projection on the image best fits the original bounding box. For this task, we adopted a Greedy Algorithm approach [42].

For each detection $D_{c,t,i}$, a regular grid of 3D points $\mathcal{G} = \{G_{x,y}\}$ corresponding to possible cylinder center-points at the XY plane is generated, $G_{x,y} = (x, y, 0)$. The grid center is the 3D reprojection of the point of the bounding box closest to the center of the image. We choose this point because, for cameras with fish-eye lenses, the furthest point from the camera of a vertical object (i.e. the feet position of a person) corresponds to the object point closest to the center of the generated image (see Fig. 6-A). The grid is defined with two parameters: the maximum distance to the center and a distance step between points. We estimate that the maximum distance to the center of the grid cannot be greater than 0.5 meters and that a distance of 0.1 meters between points is sufficient to cover the space accurately (see Fig. 6-B).

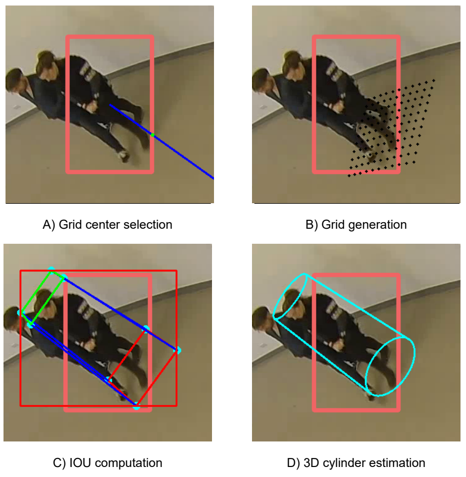


Fig. 6 Different steps for the 3D cylinder estimation. The initial red bounding box corresponds to the 2D detection and the green point in the first image corresponds to the selected grid center.

Once $\mathcal{G}$ is defined, a set of cylinders $\{C_k\}$ is generated using each grid point as the center of the cylinder at the XY plane. For each grid point, several cylinders are created, varying their diameter (min 0.5, max 1.0, step 0.15 m), and height (min 1.5, max 1.9, step 0.2 m), creating a regular sampling of the space of the cylinder $C_{c,t,i}$. The selected values have been chosen as a trade-off between the sampling density and the feet positioning error. In addition, the cylinder model is refined to better represent human shapes by reducing the upper circle radius by a constant factor (we have used 0.6 in our experiments) with respect to the lower circle. As a consequence, the cylinder becomes a truncated cone or frustrum. Since the factor is constant, it is not included into the state vector representations.

The fitting stage consists of a maximum likelihood estimator (MLE) process. A cost function is created to measure the likelihood of a given cylinder $C_j = (X, Y, H, r)$ to fit into detection $D_{c,t,i}$. The ideal cost function would be to project the cylinder outline points into the image and compute an IoU (Intersection over Union) value. For the sake of computational efficiency, the cylinder is simplified to its outer 3D cuboid, which is defined as 8 points $C_{j,k}$, $k = 1..8$, that can be projected into image points as $c_{j,k} = P_c C_{j,k}$ using homogeneous coordinates (see Fig. 6-C).

Using this cost function, the MLE estimator is obtained as the weighted sum of the grid cylinders:

$$C^*_{c,t,i} = \frac{1}{N} \sum_{j=1}^{N} IoU(b\{c_{j,k}\}, D_{c,t,i}) C_j \qquad (1)$$

where N is the total number of cylinders in the grid, spanning the three considered parameters (center, diameter and height), b is the bounding rectangle for the projected 2D points of the cuboid, and IoU is the Intersection Over Union function, obtaining the MLE estimator (see Fig. 6-D).

Although the number of occlusions is greatly reduced in BEV images from fisheye cameras, partial occlusions of the lower half of the body may appear (see Fig. 7-A). When this type of occlusions occur, the estimated cylinder $C^*_{c,t,i}$ is wrong and its upper part protrudes noticeably from the bounding box by the part furthest from the center of the image (see Fig. 7-B). This protrusion is measurable and thus an occlusion can then be detected if the salient part exceeds a certain threshold. We define the threshold value as the 10% of the distance between the points of the bounding box furthest and closest to the center of the image, which is inversely proportional to the occlusion level. Therefore, assuming that the upper point of the head, corresponding to the point of the bounding box furthest from the center of the image, is not occluded, we rebuild the cylinder from the upper center using an average human height value of 1.7 meters (see Fig. 7-D). Although the real height is likely to differ from this average height, the error produced by the occlusion is substantially reduced.

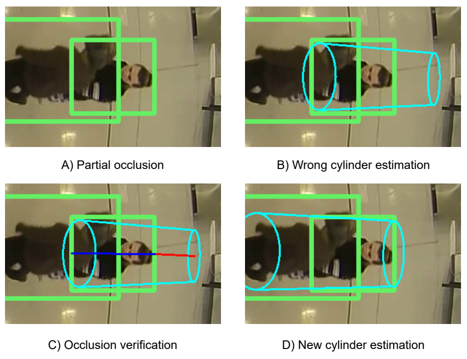

A) Partial occlusion
B) Wrong cylinder estimation
C) Occlusion verification
D) New cylinder estimation

Fig. 7 Cylinder correction from a partially-occluded detection. Note that in Figure C the distance between the points of the bounding box being furthest and closest to the image center is depicted in blue and the salient part of the cylinder is represented by the red line. The ratio between these two distances relates to the occlusion level of the detection.

As a consequence, we can then assume that the lower center point of the person corresponds to the lower center of the estimated cylinder. In Section 4.2, we compare the proposed projection method with other approaches, such as the one used in the related works [8, 21, 22, 24, 25], consisting of projecting the center of the detection bounding box. The results show that the proposed method achieves more accurate estimations and, consequently, better results in the evaluation.

3.6 Multi-Camera Detection Fusion

Once all detections $D_{c,t,i}$ have been mapped to 3D world coordinates as cylinders $C_{c,t,i}$, we search for duplicate detections in the overlap areas between two or more cameras and fuse them. The procedure consists of selecting detections that fall inside the overlapping region, and comparing them with the detections of the other camera. Two detections from different cameras are then merged if the 3D distance in the XY plane between their center points is below a certain threshold (see Fig. 8). The threshold is defined taking into account the accuracy of the calibration parameters and the average positioning error of the detections. For our experiments we selected a threshold of 0.45 meters.

From this step onwards, cylinders are no longer attached to any particular camera, and thus

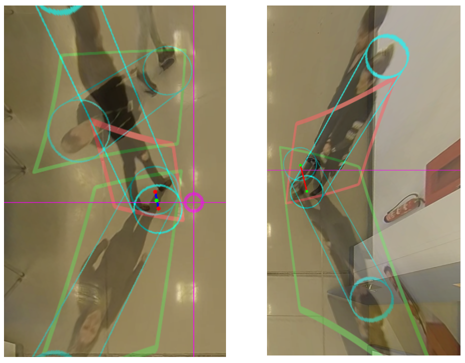


Fig. 8 Detection fusion procedure examples. In the left image, the positions of a subject captured by two overlapped cameras are fused, as the distance between the feet points is less than the defined threshold. In the right image, the cylinders belong to different subjects occluding each other to their opposite camera. In this case, the detections are not merged, as their distance is greater than the threshold.

treated as 3D objects $C_{t,i}$ in the world coordinate system.

3.7 3D People Tracking

The proposed 3D tracking algorithm is based on a data-association multi-object tracking approach [33]. The original algorithm was created to track cuboids belonging to different types of objects (vehicles and pedestrians). It is composed of two components: an online tracker and an offline post-process to smooth the trajectories and the shape of the cuboids. We modify the original algorithm taking into account the following three requirements: we only want to estimate the trajectory of people; the shape of the person is not important, it only matters that the position of their feet is precise; and the algorithm must be online. According to the latest requirement, we remove the offline cuboid smoothing component.

The remaining online tracker component consists of three stages: prediction, association and estimation. In the prediction stage, the value of the variables of each track T_k is updated based on its history using a constant-acceleration model. Such model would be useful for vehicles or even for people in scenarios where the trajectories are more steady (such as parking lots or roads). However, in scenarios such as shopping centers or airports, trajectories are more chaotic and the acceleration in one time step can vary enormously. Therefore, we decided to adopt a constant-velocity model.

During the next stage, an association matrix is created with the association likelihood between predicated tracks $\{T_k^-\}$ and detections $\{C_{t,i}\}$. The likelihood function compares the cylinders the same way as described for the multi-camera detection fusion approach (based on the distance between the feet points of the cylinders). Finally, the estimation stage updates the state of each track fusing the positioning information of the prediction and the associated cylinder following the procedure described in the original work. In order to avoid generating erroneous tracks due to false positives in the detection stage, we do not consider a track as active until it accumulates 3 or more associated detections. In the same way, to avoid removing active tracks due to false negatives in the detection stage, we keep a track as active until it has no associated detections for 5 consecutive frames.

3.8 Data Analytics

The tracks generated in the previous stage contain all the necessary information for monitoring the compliance with the main sanitary measures against COVID-19. Moreover, this information can also be used to carry out other types of tasks, such as crowd behavior understanding [43], monitoring the entry and exit of zones, the size of the waiting queues, register the most visited stands etc. In this work we focus on three tasks:

- Social distance monitoring: for this task, the Euclidean distance between the current position of each tracked subject and that of the rest is calculated and compared with the limit established by the authorities.
- Indoor capacity limitation: the number of active tracks is checked in every time step to ensure the capacity limits are not exceeded.
- Tracking of individuals who violate the sanitary measures.

An example of the results of the data analytics is presented in Fig. 9.

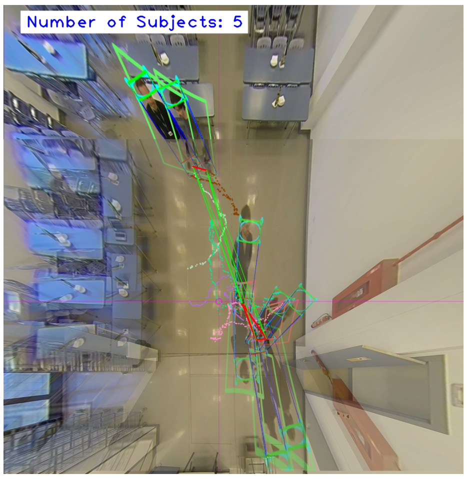


Fig. 9 Example of the results of the data analytics procedure. The interpersonal distance is measured for every pair of tracks. The red lines means that the defined safe distance is being violated. The colored dot lines represents the trajectory of every track.

4 Experiments

In this section, we conduct a series of experiments to analyze the suitability of the proposed system. Note that the goal of the experiments is not to demonstrate that we outperform the rest of the methods in terms of accuracy, as improving the quality of the person detector is not one of our main contributions. As stated in the introduction, the contributions of this work are focused on overcoming the limitations of the existing alternatives regarding the multi-camera scenarios, the monitoring range, and the scalability. Furthermore, we also present an experiment comparing the proposed 3D cylinder estimation procedure with other alternative 3D projection methods.

The server used to carry out the experiments is equipped with an NVIDIA Tesla V100 GPU and an Intel Xeon Gold 6230 CPU. Regarding the programming language, the whole system is implemented using C++.

4.1 System Performance

The accuracy of the tasks mentioned in Section 3.8 is related to the correct detection and tracking of all the individuals in the scene, and their position being accurately estimated. For this reason, to evaluate the performance of the proposed system, we focus on measuring the quality of the tracks and the accuracy of the positioning.

To measure the quality of the generated tracks, we use the metrics described in [44], developed to precisely compare different multi-object tracking methods in crowded scenes:

- Multi-object tracking accuracy ($MOTA$): evaluates the tracker performance combining the information of three sources of errors (false negatives, false positives and ID swaps).
- Mostly tracked (MT), partially tracked (PT) and mostly lost (ML) tracks: A target is mostly tracked if it is successfully tracked for at least 80% of its life span; and it is considered as mostly lost if it has been tracked for less than 20% of its total length.
- Number of fragmentations (FM): counts how many times every ground truth trajectory is interrupted (untracked).
- Fragmentation ratio (FR): relative number of fragmentations (FM/Recall).

Furthermore, to measure the accuracy of the positioning we adopt the following metrics:

- Precision (P): measures the reliability of the detections taking into account the true positives over the total positives.
- Recall (R): percentage of detected targets over the total number of targets.
- F1 Score ($F1$): harmonic mean of the precision and recall:

$$F1 = 2\frac{P \cdot R}{P + R} \qquad (2)$$

- Average Positioning Error (APE): average difference in meters between the ground truth and the estimated 3D feet positions of the people in the scene.

To evaluate the system, we consider using different public datasets [45, 46]. Nevertheless, none of them provides the intrinsic and extrinsic parameters of the involved cameras, which are necessary for the 3D projection step. Furthermore, the available datasets only cover single-camera scenarios with very limited monitoring areas. Therefore, we generate 7 sequences with different scenarios, number of cameras, heights, number of identities and levels of occlusion. The details of each

Table 1 Details of the different sequences considered for the system evaluation. For each sequence we specify the scenario, number of cameras, camera heights in meters, radius of the monitored area for each camera in meters, number of frames, number of identities, and occlusion level (from 1 to 5).

Seq	Sc	NC	Heights	Rad	Frame	IDs	Occ
1	1	2	3.3, 3.3	3.5	1610	3	1
2	1	2	3.3, 3.3	3.5	654	5	2
3	1	2	3.3, 3.3	3.5	750	5	3
4	1	2	3.3, 3.3	3.5	1083	4	4
5	1	2	3.3, 3.3	3.5	837	5	5
6	2	1	5.5	8	334	6	2
7	3	1	8.1	10	578	13	4

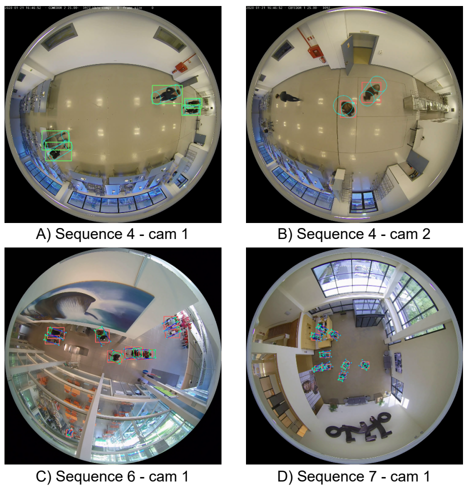


Fig. 10 Several examples of the proposed evaluation sequences.

sequence are presented in the Table 1. In addition, some examples of the different sequences are shown in Fig. 10.

For each sequence we manually annotate the identity and 3D center point on the ground plane (feet point) of all individuals. With this ground truth we extract the selected metrics. For each sequence, a monitoring region is defined. If a subject abandons this region and enters again it is considered as a new track. Therefore, the number of tracks may be greater than the number of identities. For multi-camera sequences we also evaluate the system for each camera separately to be able to analyze the impact of the multi-camera fusion procedure. The results are shown in Table 2.

From Table 2 it can be observed that, even in sequences with a high level of occlusions, the quality of the tracks (measured by the $MOTA$ metric) always remains above 90%. In the sequences with more than one camera, the achieved $MOTA$ values are very close to those obtained by processing the cameras separately. This highlights the high 3D precision obtained by 3D cylinders estimation and fusion, which allows merging detections from different views using only the lower center point of the cylinder. Furthermore, in some cases (e.g. in sequence 5), the $MOTA$ values obtained in the multi-camera sequences outperform the ones from the separate cameras. This is thanks to the multi-camera fusion procedure, where the detection errors of one camera can be corrected with the information of other overlapped cameras.

Apart from the $MOTA$ metric, the robustness of the tracks is evident by the reduced number of fragmentations. Even for sequence 5, with more than 800 frames and a high level of occlusions (see Fig. 11), only 3 fragmentations occur when the two cameras are processed together. People occluded in the furthest areas from the camera are detected by the complementary camera and vice versa. Thus, the number of fragmentations is reduced and the functional area of each camera is increased.

On the other hand, attending to the metrics considered to evaluate the precision of the positioning, it can be observed that in all the scenarios an F1 Score higher than 97% is obtained, which means that there are hardly any false positives and negatives even in the sequences with a high level of occlusions. Thus, the system is able to successfully estimate the occupancy of the monitored area. As for the APE, for sequences with a single camera it remains below 10 centimeters, while for multi-camera sequences it increases to 15-19 centimeters. This is because the calibration between cameras is not perfect and the positions estimated for the same detection from different cameras do not exactly match.

4.2 3D Projection Performance

In this section we present an experiment comparing the performance of the proposed method using three different 3D projection approaches:

Table 2 Results of the proposed method in the defined evaluation sequences. The considered metrics, described in Section 4.1, measure the quality of the generated tracks and the accuracy of the positioning. For multi-camera sequences, we also run the system for each camera separately.

Seq	Cam ID	MOTA	MT	PT	ML	FRAG	FRAG ratio	P	R	F1	APE
1	1,2	1.000	8	0	0	0	0.000	1.000	1.000	1.000	0.094
1	1	1.000	7	0	0	0	0.000	1.000	1.000	1.000	0.089
1	2	1.000	8	0	0	0	0.000	1.000	1.000	1.000	0.069
2	1,2	0.994	12	0	0	0	0.000	0.994	1.000	0.997	0.151
2	1	1.000	9	0	0	0	0.000	1.000	1.000	1.000	0.071
2	2	0.993	11	0	0	0	0.000	0.993	1.000	0.997	0.063
3	1,2	0.965	5	0	0	6	0.061	0.998	0.977	0.987	0.136
3	1	0.973	9	0	0	8	0.081	0.999	0.984	0.991	0.084
3	2	0.966	10	1	0	2	0.021	0.999	0.975	0.986	0.077
4	1,2	0.966	5	0	0	6	0.061	0.983	0.989	0.986	0.155
4	1	0.997	25	0	0	1	0.010	1.000	0.999	0.999	0.085
4	2	0.983	22	1	0	6	0.061	1.000	0.986	0.993	0.095
5	1,2	0.975	10	0	1	3	0.031	0.999	0.984	0.991	0.186
5	1	0.952	7	1	0	11	0.115	0.999	0.957	0.978	0.097
5	2	0.991	12	1	0	1	0.010	1.000	0.993	0.996	0.085
6	1	0.982	6	0	0	3	0.031	0.999	0.983	0.991	0.089
7	1	0.935	13	1	0	15	0.157	0.986	0.957	0.971	0.112

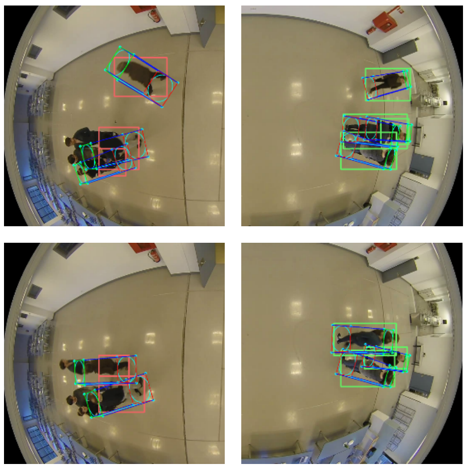


Fig. 11 Examples of occlusions in sequence 5. People occluded in the areas furthest from the camera are detected by the complementary camera. Thus, the number of fragmentations is reduced and the functional area of each camera is increased.

- Projecting the center of the detection bounding box. This is the approach followed in the related works mentioned in Section 2.1 [8, 21, 22, 24, 25].
- Projecting the point of the bounding box closest to the center of the image. As mentioned in Section 3.5, for cameras with fish-eye lenses, the furthest point from the camera of a vertical object (i.e. the feet position of a person) corresponds to the object point closest to the center of the generated image (see Fig. 6).
- Estimating the 3D cylinder. This is the proposed approach presented in Section 3.5.

We repeat the previous experiments for each projection method. The results of the experiments are presented in Table 3. In this table, we compare the metrics for measuring the positioning accuracy: precision (P), recall (R), F1 score ($F1$) and average positioning error (APE). It can be observed that the central point approach notably achieves the worst results. In addition, compared to the other approaches, with this method the

Table 3 Comparison of the performance of the proposed method using different 3D projection approaches: projecting the center of the bounding box; projecting the point closest to the center of the image; and estimating the 3D cylinder (proposed approach). S stands for the sequence number and C for the camera IDs involved in the test.

S	C	Bounding Box Center				Closest Point to Image Center				3D Cylinder Lowest Center			
		P	R	F1	APE	P	R	F1	APE	P	R	F1	APE
1	1,2	0.801	0.753	0.776	0.600	0.717	1.000	0.835	0.216	1.000	1.000	**1.000**	**0.094**
1	1	0.849	0.653	0.738	0.612	1.000	1.000	1.000	0.215	1.000	1.000	**1.000**	**0.089**
1	2	0.893	0.710	0.791	0.453	1.000	1.000	1.000	0.273	1.000	1.000	**1.000**	**0.069**
2	1,2	0.827	0.732	0.776	0.572	0.818	0.995	0.898	0.388	0.994	1.000	**0.997**	**0.151**
2	1	0.687	0.445	0.540	0.618	1.000	1.000	1.000	0.251	1.000	1.000	**1.000**	**0.071**
2	2	0.809	0.463	0.589	0.526	1.000	1.000	**1.000**	0.355	0.993	1.000	0.997	**0.063**
3	1,2	0.765	0.705	0.734	0.481	0.840	0.970	0.900	0.301	0.998	0.977	**0.987**	**0.136**
3	1	0.880	0.476	0.617	0.524	0.987	0.982	0.985	0.287	0.999	0.984	**0.991**	**0.084**
3	2	0.892	0.627	0.736	0.481	0.998	0.972	0.985	0.326	0.999	0.975	**0.986**	**0.077**
4	1,2	0.767	0.741	0.754	0.471	0.798	0.986	0.882	0.297	0.983	0.989	**0.986**	**0.155**
4	1	0.895	0.622	0.734	0.493	0.990	0.998	0.994	0.268	1.000	0.999	**0.999**	**0.085**
4	2	0.763	0.556	0.643	0.455	0.978	0.985	0.982	0.304	1.000	0.986	**0.993**	**0.095**
5	1,2	0.873	0.751	0.807	0.504	0.860	0.970	0.912	0.395	0.999	0.984	**0.991**	**0.186**
5	1	0.903	0.505	0.647	0.569	0.997	0.945	0.971	0.258	0.999	0.957	**0.978**	**0.097**
5	2	0.811	0.484	0.606	0.521	0.992	0.994	0.993	0.330	1.000	0.993	**0.996**	**0.085**
6	1	0.907	0.830	0.867	0.331	0.978	0.974	0.976	0.332	0.999	0.983	**0.991**	**0.089**
7	1	0.927	0.888	0.907	0.474	0.982	0.953	0.967	0.272	0.986	0.957	**0.971**	**0.112**

number of false positives and negatives increases, reflected in the decrease in precision and recall. Finally, if we compare the other two methods, the proposed approach achieves the best result by far, reducing the average positioning error by more than 4 times in most cases. It can be observed that, when using the closest point to the image center, the precision for multi-camera sequences worsens. These false positives are caused because the positioning using this method is not accurate enough to fuse the detections of the overlapped cameras. Several examples of the positions estimated by the evaluated approaches are shown in Fig. 12.

5 Discussion

In this section we want to analyze the results obtained in the experiments presented on the previous section and highlight the strengths and weaknesses of the proposed method, comparing it with other approaches mentioned in the related work. In [24] and [25], the authors declare that they use overhead datasets for training and testing. However, we were unable to find these datasets available online for comparison. The rest of the related works use images with side or frontal perspectives, so it is not possible to make a fair comparison with them. Furthermore, none of the test datasets they use include the necessary calibration parameters to be able to carry out the projection. Finally, none of the mentioned methods is publicly available, so we cannot make a comparison using our test sequences either. For this reason, we focus on comparing the proposed projection method with the projection method used in all other related works. All the mentioned methods that perform projection [8, 21, 22, 24, 25] use the central point of the bounding box to estimate the position of the subject. As commented in the previous section, from the results of Table 3, it can be observed that our method outperforms the the method based on central point projection,

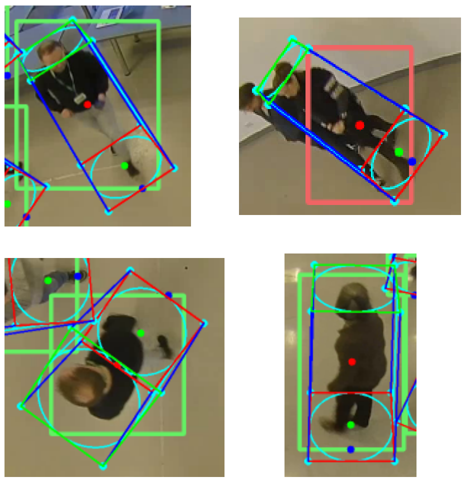


Fig. 12 Several examples of the positions estimated by the 3D projection approaches compared in Section 4.2. The red point corresponds to the center of the detection bounding box, the blue point to the point closest to the center of the image, and the green point to the proposed method.

decreasing the average positioning error by more than 4 times in most of the cases.

Main drawback is that estimating the 3D cylinder requires additional computing time. More specifically, for the hardware used in the evaluation (Intel Xeon Gold 6230), the estimation time for each cylinder is 0.6 ms. It is a despicable amount of time but it grows in proportion to the number of detections. Nevertheless, this procedure would be easily parallelizable using multi-threading on CPU or GPU, since the estimation of each cylinder is independent of the rest.

Regarding the size of the monitored area, in [24] and [25] the authors do not give any details about this topic. Analyzing the data and images provided in the articles, we estimate that they cover an area with an approximate radius of 5 meters for a fisheye camera placed 6 meters high. If we compare this setup with the one of our evaluation sequence 6 (see Table 1), it can be observed that, even though the camera is positioned at a lower height, the radius of the area covered (8 meters) by our method is much higher.

Another strong point of the proposed method is its ease of adaptation to new environments. As we discussed in previous sections, the system is capable of working with cameras positioned at different heights, thanks to the fact that the fusion of multi-camera detections and tracking is carried out taking into account the position in real coordinates and not the 2D bounding box of the detection. This makes the method adaptable to any indoor environment with the only requirements that there is a ceiling where to place the cameras, that the cameras are placed perpendicular to the ground plane and that there is a small overlap region between two consecutive cameras.

Finally, the proposed system is also easily scalable, allowing new cameras to be added at any time. The newly added cameras only need to be calibrated to obtain the intrinsic and extrinsic parameters referenced to the coordinate origin of the global system.

6 Conclusions

The aim of this work was to create a system capable of monitoring people in large infrastructures, especially to guarantee compliance with the health measures imposed by COVID-19. In order to algorithmically assess compliance with some of these measures, such as maintaining social distance, a precise position estimation of the subjects is necessary and complete occlusions must be avoided. Therefore, we decided to tackle the problem using people detection from an overhead perspective. This is yet an unexplored topic and the few solutions proposed only work in single camera scenarios and cover a very limited area.

To overcome these limitations, we present a multi-camera BEV people flow monitoring system, capable of extracting reliable real-time performance indicators in extremely large infrastructures, such as airports or shopping centers. The proposed system breaks with the traditional pipeline, applying the projection step just after the detection stage. This modification allows tracking the subjects uninterruptedly all over the monitored area using just a single tracker instance and using multiple cameras installed at different heights. Furthermore, we present a novel 3D projection and multi-camera fusion procedure. It estimates the best fitting 3D cylinder for each detected bounding box and fuses the cylinders of the overlapping regions of the camera views that belong to the same person. This corrects possible occlusion problems and allows us to expand the useful range of the cameras.

Conducted experiments, presented in Section 4, demonstrate that the proposed system is suitable for real-time sanitary-measures-control applications, such as guaranteeing a safe interpersonal distance, respecting the indoor capacity limitations, identifying the most crowded time intervals or tracking subjects who violate the established measures. Furthermore, the proposed projection approach achieves an average positioning error below 0.2 meters, with an improvement of more than 4 times compared to other methods.

Future work will focus on extending the application of the system to other tasks such as subject re-identification. For this task, the system should be able to fuse the information of overhead and frontal cameras in order to identify the subject using facial recognition and track it throughout infrastructure. The strengths of the proposed method lie in its ability to monitor large areas in an efficient and scalable manner. It could be used in other applications that require this capacity, such as for monitoring traffic in cities or dangerous vehicles that have committed an infraction. We also think that it suitable for monitoring certain sports in which the playing field is very wide, such as football or rugby, both for tracking the players and the ball. Furthermore, we will also work on improving the accuracy of the detection network to reduce the number of false positives and negatives and improve the performance of the entire system. Finally, we will also focus on improving the data analytics logic to incorporate also temporal information for the distance monitoring, using the computed subject trajectories.

Declarations

Conflicts of interest The authors declare that they have no conflicts of interest.

References

[1] Chen, S.-C.: Multimedia research for response and management of covid-19 and future pandemics. IEEE MultiMedia **28**(1), 5–6 (2021). https://doi.org/10.1109/MMUL.2021.3063011

[2] Rahmani, A.M., Mirmahaleh, S.Y.H.: Coronavirus disease (covid-19) prevention and treatment methods and effective parameters: A systematic literature review. Sustainable Cities and Society **64**, 102568 (2021). https://doi.org/10.1016/j.scs.2020.102568

[3] Agarwal, N., Meena, C.S., Raj, B.P., Saini, L., Kumar, A., Gopalakrishnan, N., Kumar, A., Balam, N.B., Alam, T., Kapoor, N.R., Aggarwal, V.: Indoor air quality improvement in covid-19 pandemic: Review. Sustainable Cities and Society **70**, 102942 (2021). https://doi.org/10.1016/j.scs.2021.102942

[4] Su, X., Gao, M., Ren, J., Li, Y., Dong, M., Liu, X.: Face mask detection and classification via deep transfer learning. Multimedia Tools and Applications (2021). https://doi.org/10.1007/s11042-021-11772-5

[5] Singh, S., Ahuja, U., Kumar, M., Kumar, K., Sachdeva, M.: Face mask detection using yolov3 and faster r-cnn models: Covid-19 environment. Multimedia Tools and Applications, 19753–19768 (2021). https://doi.org/10.1007/s11042-021-10711-8

[6] Sun, C., Zhai, Z.: The efficacy of social distance and ventilation effectiveness in preventing covid-19 transmission. Sustainable Cities and Society **62**, 102390 (2020). https://doi.org/10.1016/j.scs.2020.102390

[7] Thu, T.P.B., Ngoc, P.N.H., Hai, N.M., *et al.*: Effect of the social distancing measures on the spread of covid-19 in 10 highly infected countries. Science of the Total Environment **742**, 140430 (2020)

[8] Rezaei, M., Azarmi, M.: Deepsocial: Social distancing monitoring and infection risk assessment in covid-19 pandemic. Applied Sciences **10**(21) (2020). https://doi.org/10.3390/app10217514

[9] Uras, M., Cossu, R., Ferrara, E., Liotta, A., Atzori, L.: Pma: A real-world system for people mobility monitoring and analysis based on wi-fi probes. Journal of Cleaner Production **270**, 122084 (2020). https://doi.org/10.1016/j.jclepro.2020.122084

[10] Bian, S., Zhou, B., Bello, H., Lukowicz, P.: A wearable magnetic field based proximity sensing system for monitoring covid-19 social distancing. In: Proceedings of the 2020 International Symposium on Wearable Computers. ISWC 20, pp. 22–26. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3410531.3414313

[11] Shao, Z., Cheng, G., Ma, J., Wang, Z., Wang, J., Li, D.: Real-time and accurate uav pedestrian detection for social distancing monitoring in covid-19 pandemic. IEEE Transactions on Multimedia, 1–1 (2021). https://doi.org/10.1109/TMM.2021.3075566

[12] Sathyamoorthy, A.J., Patel, U., Savle, Y.A., Paul, M., Manocha, D.: COVID-Robot: Monitoring Social Distancing Constraints in Crowded Scenarios (2020). http://arxiv.org/abs/2008.06585

[13] Montero, D., Unzueta, L., Goenetxea, J., Aranjuelo, N., Loyo, E., Otaegui, O., Nieto, M.: Multi-stage dynamic batching and on-demand i-vector clustering for cost-effective video surveillance. In: VISAPP. VISIGRAPP 2021. SciTePress, ??? (2021)

[14] Ali, W., Tian, W., Swati, S., Iradukunda, D., Khan, A.: Classical and modern face recognition approaches: a complete review. Multimedia Tools and Applications **80**, 1–56 (2021). https://doi.org/10.1007/s11042-020-09850-1

[15] Chu, S.-L., Chen, C.-F., Zheng, Y.-C.: Cfsm: a novel frame analyzing mechanism for real-time face recognition system on the embedded system. Multimedia Tools and Applications (2021)

[16] Wang, Y., Hu, S., Wang, G., Chen, C., Pan, Z.: Multi-scale dilated convolution of convolutional neural network for crowd counting. Multimedia Tools and Applications **79** (2020). https://doi.org/10.1007/s11042-019-08208-6

[17] Zou, Z., Li, C., Zheng, Y., Xu, S.: Two stages double attention convolutional neural network for crowd counting. Multimedia Tools and Applications **79** (2020). https://doi.org/10.1007/s11042-020-09541-x

[18] Chen, L., Wang, G., Hou, G.: Multi-scale and multi-column convolutional neural network for crowd density estimation. Multimedia Tools and Applications **80**, 6661–6674 (2021)

[19] Su, J., He, X., Qing, L., Niu, T., Cheng, Y., Peng, Y.: A novel social distancing analysis in urban public space: A new online spatio-temporal trajectory approach. Sustainable Cities and Society **68**, 102765 (2021). https://doi.org/10.1016/j.scs.2021.102765

[20] Shorfuzzaman, M., Hossain, M.S., Alhamid, M.F.: Towards the sustainable development of smart cities through mass video surveillance: A response to the covid-19 pandemic. Sustainable Cities and Society **64**, 102582 (2021). https://doi.org/10.1016/j.scs.2020.102582

[21] Punn, N.S., Sonbhadra, S.K., Agarwal, S.: Monitoring COVID-19 social distancing with person detection and tracking via fine-tuned YOLO v3 and Deepsort techniques (2020). http://arxiv.org/abs/2005.01385

[22] Yang, D., Yurtsever, E., Renganathan, V., Redmill, K.A., Ümit Özgüner: A Vision-based Social Distancing and Critical Density Detection System for COVID-19 (2020). http://arxiv.org/abs/2007.03578

[23] Nodehi, H., Shahbahrami, A.: Multi-metric re-identification for online multi-person tracking. IEEE Transactions on Circuits and Systems for Video Technology, 1–1 (2021). https://doi.org/10.1109/TCSVT.2021.3059250

[24] Ahmed, I., Ahmad, M., Rodrigues, J., Jeon, G., Din, S.: A deep learning-based social distance monitoring framework for covid-19. Sustainable Cities and Society **65**, 102571 (2020). https://doi.org/10.1016/j.scs.2020.102571

[25] Ahmed, I., Ahmad, M., Jeon, G.: Social distance monitoring framework using deep

learning architecture to control infection transmission of covid-19 pandemic. Sustainable Cities and Society **69**, 102777 (2021). https://doi.org/10.1016/j.scs.2021.102777

[26] Nguyen, C.T., Saputra, Y.M., Huynh, N.V., Nguyen, N.-T., Khoa, T.V., Tuan, B.M., Nguyen, D.N., Hoang, D.T., Vu, T.X., Dutkiewicz, E., Chatzinotas, S., Ottersten, B.: A comprehensive survey of enabling and emerging technologies for social distancing—part i: Fundamentals and enabling technologies. IEEE Access **8**, 153479–153507 (2020). https://doi.org/10.1109/ACCESS.2020.3018140

[27] Nguyen, C.T., Saputra, Y.M., Van Huynh, N., Nguyen, N.-T., Khoa, T.V., Tuan, B.M., Nguyen, D.N., Hoang, D.T., Vu, T.X., Dutkiewicz, E., Chatzinotas, S., Ottersten, B.: A comprehensive survey of enabling and emerging technologies for social distancing—part ii: Emerging technologies and open issues. IEEE Access **8**, 154209–154236 (2020). https://doi.org/10.1109/ACCESS.2020.3018124

[28] Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. CoRR **abs/1804.02767** (2018)

[29] 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), pp. 3645–3649 (2017). https://doi.org/10.1109/ICIP.2017.8296962

[30] Ramadass, L., Arunachalam, S., Sagayasree, Z.: Applying deep learning algorithm to maintain social distance in public place through drone technology. Int. J. Pervasive Comput. Commun. **16**, 223–234 (2020)

[31] Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y.M.: YOLOv4: Optimal Speed and Accuracy of Object Detection (2020). http://arxiv.org/abs/2004.10934

[32] Bewley, A., Ge, Z., Ott, L., Ramos, F., Upcroft, B.: Simple online and realtime tracking. 2016 IEEE International Conference on Image Processing (ICIP) (2016). https://doi.org/10.1109/icip.2016.7533003

[33] Montero, D., Aranjuelo, N., Senderos, O., Nieto, M.: Bev object tracking for lidar-based ground truth generation. In: 2019 27th European Signal Processing Conference (EUSIPCO) (2019)

[34] Ahmed, I., Adnan, A.: A robust algorithm for detecting people in overhead views. Cluster Computing **21**, 1–22 (2018). https://doi.org/10.1007/s10586-017-0968-3

[35] Ahmed, I., Ahmad, A., Piccialli, F., Sangaiah, A.K., Jeon, G.: A robust features-based person tracker for overhead views in industrial environment. IEEE Internet of Things Journal **5**(3), 1598–1605 (2018). https://doi.org/10.1109/JIOT.2017.2787779

[36] Ahmad, J., Larijani, H., Emmanuel, R., Mannion, M., Javed, A.: An intelligent real-time occupancy monitoring system using single overhead camera. In: Proceedings of the 2018 Intelligent Systems Conference, vol. 2, pp. 957–969 (2019). https://doi.org/10.1007/978303001057771

[37] Fernandez-Rincon, A., Fuentes-Jimenez, D., Losada-Gutierrez, C., Marron-Romera, M., Luna, C.A., Macias-Guarasa, J., Mazo, M.: Robust People Detection and Tracking from an Overhead Time-of-Flight Camera. In: 12th International Conference on Computer Vision Theory and Applications., Porto, Portugal, pp. 556–564 (2017). https://doi.org/10.5220/0006169905560564

[38] Aranjuelo, N., García, S., Loyo, E., Unzueta, L., Otaegui, O.: Key strategies for synthetic data generation for training intelligent systems based on people detection from omnidirectional cameras (in press). Computers & Electrical Engineering (2021)

[39] Zoph, B., Cubuk, E.D., Ghiasi, G., Lin, T.-Y., Shlens, J., Le, Q.V.: Learning data augmentation strategies for object detection. In: European Conference on Computer Vision, pp. 566–583 (2020). Springer

[40] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C.L., Dollár, P.: Microsoft COCO: Common Objects in Context (2015). http://arxiv.org/abs/1405.0312

[41] Zhang, P., Zhong, Y., Li, X.: Slimyolov3: Narrower, faster and better for real-time UAV applications. CoRR **abs/1907.11093** (2019)

[42] Annu Malik, M.V.S. Anju Sharma: Greedy Algorithm. In: International Journal of Scientific and Research Publications, vol. 3 (2013)

[43] Li, Y.: A deep spatiotemporal perspective for understanding crowd behavior. IEEE Transactions on Multimedia **20**(12), 3289–3297 (2018). https://doi.org/10.1109/TMM.2018.2834873

[44] Dendorfer, P., Rezatofighi, H., Milan, A., Shi, J., Cremers, D., Reid, I., Roth, S., Schindler, K., Leal-Taixé, L.: MOT20: A benchmark for multi object tracking in crowded scenes (2020). http://arxiv.org/abs/2003.09003

[45] Scheck, T., Seidel, R., Hirtz, G.: Learning from theodore: A synthetic omnidirectional top-view indoor dataset for deep transfer learning. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 943–952 (2020)

[46] Li, S., Tezcan, M.O., Ishwar, P., Konrad, J.: Supervised people counting using an overhead fisheye camera. In: 2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pp. 1–8 (2019). https://doi.org/10.1109/AVSS.2019.8909877

4.4 Multi-Stage Dynamic Batching and On-Demand I-Vector Clustering for Cost-effective Video Surveillance

- **Authors:** David Montero and Luis Unzueta and Jon Goenetxea and Nerea Aranjuelo and Estíbaliz Loyo and Oihana Otaegui and Marcos Nieto

- **Booktitle:** Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)

- **Year:** 2021

- **Publisher:** SciTePress

Multi-Stage Dynamic Batching and On-Demand I-Vector Clustering for Cost-Effective Video Surveillance

David Montero, Luis Unzueta, Jon Goenetxea, Nerea Aranjuelo, Estibaliz Loyo, Oihana Otaegui, Marcos Nieto
Vicomtech, Mikeletegi 57, 20009 Donostia-SanSebastian, Spain
{*dmontero, lunzueta, jgoenetxea, naranjuelo, eloyo, ootaegui, mnieto*}*@vicomtech.org*

Keywords: Face Recognition, Face Clustering, Video-Surveillance

Abstract: In this paper, we present a cost-effective Video-Surveillance System (VSS) for face recognition and online clustering of unknown individuals at large scale. We aim to obtain Performance Indicators (PIs) for people flow monitoring in large infrastructures, without storing any biometric information. For this purpose, we focus on how to take advantage of a central GPU-enabled computing server, connected to a set of video-surveillance cameras, to automatically register new identities and update their descriptive data as they are re-identified. The proposed method comprises two main procedures executed in parallel. A Multi-Stage Dynamic Batching (MSDB) procedure efficiently extracts facial identity vectors (i-vectors) from captured images. At the same time, an On-Demand I-Vector Clustering (ODIVC) procedure clusters the i-vectors into identities. This clustering algorithm is designed to progressively adapt to the increasing data scale, with a lower decrease in its effectiveness compared to other alternatives. Experimental results show that ODIVC achieves state-of-the-art results in well-known large scale datasets and that our VSS can detect, recognize and cluster in real time faces coming from up to 40 cameras with a central off-the-shelf GPU-enabled computing server.

1 INTRODUCTION

In recent years, there has been a growing interest in obtaining a detailed operational experience of people flow monitoring in large infrastructures, by means of Performance Indicators (PIs), such as ”waiting times”, ”process throughput”, ”queue length overrun” and ”area occupancy” (Mayer et al., 2015). Current non-cooperative solutions are typically integrated in the terminal infrastructure or use data from existing processes, such as entrance pass scans or mobile devices and Wi-Fi signals, from which PIs can be derived. As stated in (Mayer et al., 2015), for this kind of applications, computer-vision-based facial recognition technology has been used mainly as a source to infer ”waiting times”, by comparing biometric information from entry and exit points.

Our motivation is to enhance and extend the applicability of computer-vision-based facial recognition technology to more cases than the estimation of ”waiting times” in a specific area of a large infrastructure. More specifically, our goal is to build a face recognition-based solution from which all the PIs can be derived for the whole infrastructure, trying to simplify as much as possible the required hardware setup, and with a better handling of data so that privacy issues can be avoided. This requires building an accurate and efficient face recognition system that can manage large-scale data, without storing the biometric information of the individuals. Deep Neural Network (DNN)-based large-scale clustering approaches are promising methodologies for this purpose (Wang et al., 2019; Shi et al., 2018; Otto et al., 2018).

Deploying computer vision algorithms to build a cost-effective Video-Surveillance System (VSS) is challenging. The latest trends rely on distributed computing infrastructures, based on cloud (Lim et al., 2018), fog (Nasir et al., 2018) and/or edge computing paradigms (Chen et al., 2019). These solutions are capable of processing multiple sensor data streams more efficiently at a bigger scale, compared to the traditional centralized infrastructures (Tsakanikas and Dagiuklas, 2017). The main open questions in the design of a VSS (distributed or centralized) are what equipment should be used and how to take advantage of the available computing capabilities. This work focuses on the second question when an off-the-shelf GPU computing server is considered. This solution directly addresses the centralized infrastructure case, but is likely extendable to distributed infrastructures.

These are the main contributions of our work:

- A cost-effective VSS for face recognition and online clustering of unknown individuals at large scale, without storing their biometric information, to obtain PIs for people flow monitoring.
- A Multi-Stage Dynamic Batching (MSDB) procedure to efficiently extract face attributes and i-vectors from captured images. This includes MB-MTCNN, a multi-batch version of a Multi-Task Cascaded Convolutional Network (MTCNN) (Zhang et al., 2016a), and an efficient dynamic batching strategy for the processing of dynamic lists of facial images.
- An On-Demand I-Vector Clustering (ODIVC) algorithm, designed to progressively adapt to the data scale, with a lower decrease on its effectiveness compared to state-of-the-art alternatives.

The paper is organized as follows. Section 2 presents the related work. Section 3 describes the proposed VSS, followed by the computation improvements and optimizations in section 4. Section 5 provides experimental results that show the potential of our approach. Finally, section 6 concludes the paper.

2 RELATED WORK

The main challenges in our context are two: (A) how to cluster faces by identity, using an online and unsupervised approach, with the necessary accuracy and scalability; and (B) how to build a cost-effective VSS to deploy this technology.

2.1 Face Clustering

In recent years, face clustering in large-scale unconstrained scenarios has become a major challenge. The huge number of faces and the intra-class changes due to environmental variations (e.g., pose, illumination, occlusions, resolution, noise) lead to complex distributions of face representations. This makes it unsuitable to apply classic algorithms like K-Means (Lloyd, 1982), which tend to generate similar sized clusters, or spectral clustering (Shi and Malik, 2000).

State-of-the-art methodologies combine DNN-based face recognition models to extract i-vectors with clustering algorithms that can group them in distinguishable identities, despite the intra-class variability. In (Shi et al., 2018) the ConPaC algorithm is proposed, which is based on the estimation of an adjacency matrix using pairwise similarities between i-vectors. In (Wang et al., 2019) GCN is presented, where a DNN decides which pairs of nodes should be linked. In (Lin et al., 2018) an Agglomerative Hierarchical Clustering approach is adopted, considering the distance in the embedded space and the dissimilarity between groups of faces. In (Otto et al., 2018) an approximate rank-order clustering is presented, which predicts whether a node should be linked to its k Nearest Neighbors (kNN), and merges all linked pairs.

Nevertheless, these methodologies use offline algorithms. They process entirely the gathered data every time a new i-vector arrives with increasing computational cost. In addition, most of them suffer from scalability problems. For instance, the complexity of ConPaC can scale up to $O(TN^3)$, where N is the number of i-vectors and T the number of iterations. (Wang et al., 2019) and (Otto et al., 2018) reduce the complexity using kNN graphs to reduce the number of comparisons, but the cost is still too high for considering them for online applications.

To overcome these problems, we present On-Demand I-Vector Clustering algorithm (ODIVC). It is suitable for large-scale real-time applications, as it is devised to dynamically adapt to the inclusion of data samples without repeating the whole process.

2.2 Computing Infrastructures for Automated Video-surveillance

New automated surveillance systems incorporate modern video sensors capable of capturing high definition footage and DNN-based processing pipelines in real-time. These systems frequently require large computational resources and large storage size. Both requirements could be resolved by integrating the VSS in a distributed computing infrastructure, relying on cloud, fog and/or edge computing paradigms. However, this introduces new challenges to be addressed (Tsakanikas and Dagiuklas, 2017).

A VSS relying on cloud computing (Lim et al., 2018) needs to take into account the latency and extra communication cost introduced between the sensors and the cloud infrastructure. Furthermore, it needs to deal with the latency introduced in IP networks, which is not only large but also fluctuating. Fog computing is a complementary technology to cloud computing, as it extends cloud capabilities to the edge of the network, comprising devices that have enough power to perform non-trivial computational tasks. A representative example of fog-based VSS is presented in (Nasir et al., 2018). Finally, edge computing refers to transferring computational and storage capacities from data centers to the video sensors, minimizing the network latency. Its application in a VSS requires the usage of special hardware and software close to the video sensors (Chen et al., 2019).

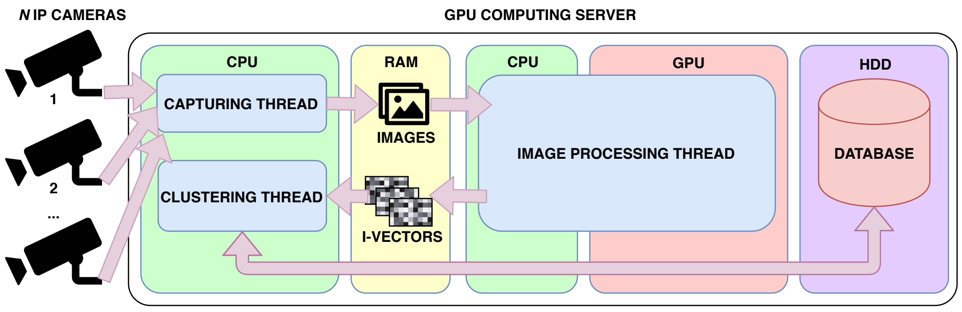


Figure 1: Overall processing architecture and data flow diagram of the proposed VSS.

As stated in (Tsakanikas and Dagiuklas, 2017), a promising approach for a VSS could be a distributed architecture, where certain characteristics of each approach are utilized to maximize its efficiency. In any case, effectively deploying DNN-based technology in such kind of equipment to build a cost-effective VSS, is a challenge to be addressed.

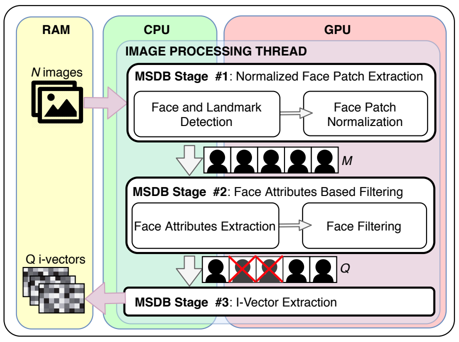
3 PROPOSED GPU COMPUTING SERVER-BASED VSS

3.1 Capturing Thread

The capturing thread is in charge of three tasks: camera connection handling, stream decoding and image acquisition. The effective camera management involves not only creating the connections when the system startups, but also periodically checking whether they are still working and reconnecting in case of a lost connection. Image streaming uses a compression method to reduce the network overload (e.g., H.264, H.265, etc) (Jankar and Shah, 2017), so a decoding phase is needed before feeding the images into the processing thread. This can cause a delay in the results. To reduce this effect, the capturing thread decodes each image stream separately. Thus, the current image is ready when the image processing thread needs to get the next frame. The computational cost of the capturing thread is low compared with the rest and it only uses a small amount of the CPU for frame decoding, mask applying, and connection checking.

3.2 Image Processing Thread

The image processing thread applies the DNN-based face analysis algorithms to the captured images,



RAM
CPU
GPU
IMAGE PROCESSING THREAD
N images
MSDB Stage #1: Normalized Face Patch Extraction
Face and Landmark Detection
Face Patch Normalization
M
MSDB Stage #2: Face Attributes Based Filtering
Face Attributes Extraction
Face Filtering
Q
Q i-vectors
MSDB Stage #3: I-Vector Extraction

Figure 2: MSDB procedure in the image processing thread.

in order to get the required i-vectors for the re-identification. It consumes most of the computation time of the entire VSS (90-95%). Thus, to improve the global performance, the system exploits the GPU resources for the DNN inferences, using the CPU for minor tasks like processing flow control, image cropping and managing patch lists (Figure 2).

This procedure has three sequential stages: 1) normalized face patch extraction, 2) facial attribute-based filtering, and 3) i-vector extraction.

The first stage detects all the faces present in the input image list and their facial landmarks, which are used for the face patch normalization following the method described in (Wang et al., 2018). For an efficient detection of the facial regions and landmarks in the *N* images from the capture process we propose a multi-batch version of MTCNN (MB-MTCNN), described in section 4.1, with a batch size equal to the number of images (i.e. *N*). This stage generates a list of *M* face regions with a set of five facial landmarks.

The VSS captures images from uncontrolled environments, so the detected faces may have different orientations, lighting conditions, partial occlusions, etc., which can negatively influence in the i-vector extraction, reducing the accuracy of the re-identification process (Tan and Triggs, 2010). It is well known that the impact of these factors can be reduced by preprocessing the face patches before extracting the i-vector, as stated in (Chaudhari and Kale, 2010).

Thus, the second stage checks if the facial patch is suitable for re-identification using a set of automatically detected attributes and previously defined filtering considerations. In our context, we use head orientation and a set of angle thresholds for the filtering process, but other descriptive attributes could also be considered (e.g. age, gender, ethnic group, etc.). These attributes are estimated with Multi-Task Cascaded Convolutional Networks (TCDCN) (Zhang et al., 2016b). All the patches that do not match the established rules are removed.

Finally, the last stage extracts the i-vectors from the list of filtered face patches given by the filtering stage. For our experiments we use a DNN model based on ResNet100 architecture (He et al., 2016) with ArcFace loss (Deng et al., 2019). We use this architecture, despite its complexity, because we need to generate the i-vectors as robust as possible in order to get highly-reliable PIs. The DNN inferences applied to the dynamic lists of cropped facial images are made following efficient dynamic batching procedures, as explained in section 4.2.

3.3 Clustering Thread

This thread analyzes the incoming i-vectors to automatically register new people and update their descriptive data with new samples as they are re-identified, in an unsupervised way. Algorithm 1 shows our proposed ODIVC procedure for this task.

The identities database is represented with four lists: **C**, containing the representative i-vector (centroid) of each registered identity; **SC**, containing the sum of the candidates i-vectors of each registered identity; **uid**, with the unique identification numbers for those identities; and **nm**, with the number of detections matched with each identity. Thus, the database is expressed as <**C**, **SC**, **uid**, **nm**>. The new identity candidates do not get a unique identification number until they have reached a certain number of matched detections. This is done to filter erroneous new identity candidates, created with unsuitable faces passed through the filters of the detection stage.

When the VSS is launched, the identities database is empty. Therefore, the first incoming i-vector will be used as the representative i-vector of the first identity candidate in the database. From then on, every new incoming i-vector is compared with the centroid of every registered identity. The comparison is done by computing the cosine similarity, as the employed face recognition model was trained to work with this metric (Deng et al., 2019), but other similarity measures may be considered. To speed up the computation of the cosine similarity, every incoming i-vector and centroids are normalized, so that only the dot product between them needs to be computed. Besides, the calculation of the dot product is parallelized. Once the cosine similarity is extracted, the algorithm searches for the best candidate above a predefined minimum threshold ($thresh_{low}$). If the selected identity has more than nm_{min} members, then the centroid is considered robust enough for using $thresh_{low}$ and the i-vector is matched to that identity. Otherwise, the similarity measure must be higher than $thresh_{high}$. This search-

Algorithm 1: On-Demand I-Vector Clustering

```
Input: new i-vector u, database of identities
       <C, SC, uid, nm> (initially empty),
       low similarity threshold $thresh_{low}$
       [-1,1], high similarity threshold
       $thresh_{high}$ [-1,1], minimum cluster
       members for low threshold $nm_{min}$
Output: Updated DB <C, SC, uid, nm>

$\mathbf{u}_{norm} = ||\mathbf{u}||_2$
$\mathbf{sim} = \mathbf{u}_{norm} \cdot \mathbf{C}$
$match$ = -1, $sim_{best}$ = -1
if $sizeof(\boldsymbol{R}) > 0$ then
    $idx_{best} = maxIdx(\boldsymbol{sim})$
    $sim_{best} = \mathbf{sim}[idx_{best}]$
end
while $sim_{best} > thresh_{low}$ do
    if $\boldsymbol{nm}[idx_{best}] > nm_{min}$ or
      $sim_{best} > thresh_{high}$ then
        $match = idx_{best}$
        break
    end
    $\mathbf{sim}[idx_{best}]$ = -1
    $idx_{best} = maxIdx(\boldsymbol{sim})$
    $sim_{best} = \mathbf{sim}[i_{best}]$
end
if $match > -1$ then
    $\mathbf{nm}[match] = \mathbf{nm}[match] + 1$
    $\mathbf{SC}[match] = \mathbf{SC}[match] + \mathbf{u}_{norm}$
    $\mathbf{C}[match] = ||\mathbf{SC}[match]||_2$
else
    $uid_{new}$ = generateNewUID()
    Append $\mathbf{u}_{norm}$ to C & SC, $uid_{new}$ to uid
      and 1 to nm
end
return <C, SC, uid, nm>
```

ing process is repeated until there is a match or until the best similarity measure is lower than $thresh_{low}$. If there is a match between the incoming i-vector and a registered identity, the new i-vector is used to update the sum and the normalized centroid of the identity, as shown in the algorithm. Otherwise, it becomes directly the representative centroid of a new identity. The time complexity of the algorithm is $O(MC)$, where M is the i-vector dimensionality and C is the number of active identities.

Alternative registration and updating strategies might be considered, for example, a set of i-vectors per identity represented by their median for the comparisons (which theoretically is more robust to outliers than the running average). However, we register each individual with only a single i-vector and update it with the running average strategy. This ensures the minimal amount of information is stored, which is critical to handle large scales. In addition, according to our experiments, it is less susceptible to noise, i.e. erroneous faces that have reached the clustering stage.

The RAM memory is the fastest option to store these data but it is not safe to system failures (e.g. power supply failures). To avoid data loss, the system also stores the registration information in a persistent database, to use it as an information cache if the system fails or needs to be restarted. Additional spatiotemporal constraints related to characteristics of the infrastructure (e.g. one-way zone-to-zone doors, or one-way exit doors that assure that an individual has left the infrastructure at least for a certain time, etc.), allow reducing the number of comparisons and, in consequence, potential erroneous matches.

4 MSDB OPTIMIZATIONS

4.1 Multi-Batch MTCNN

To improve the computation performance of the face and landmark detections from full-images in the first stage of MSDB, we adopt and modify the MTCNN method (Zhang et al., 2016a), resulting in the proposed MB-MTCNN approach, which includes multi-batch processing and a series of parallelization strategies on each stage of MTCNN.

We chose MTCNN as the detection network because of some remarkable characteristics. First, it performs a multi-scale search, generating candidate face regions at several image scales. This allows us to parallelize the generation of candidates by image scale, and also allows configuring the scales to optimize the inference time. Furthermore, it provides face regions and landmarks in a single forward pass, sharing features between both kinds of detections and saving inference time. Finally, it needs a small size in memory, allowing us to use more limited hardware resources.

MTCNN has three stages. The first one processes the input image at different scales, and generates facial region candidates using a convolutional neural network (CNN) called *Proposal Network* (P-Net). The second stage refines the candidate regions using a CNN called *Refinement Network* (R-Net). The third stage refines the facial regions generated by R-Net, and detects five landmarks inside each of them using a CNN called *Output-Network* (O-Net).

In MB-MTCNN, the key factor to accelerate this process is the inclusion of parallel while loops, built upon flexible and expressive control-flow primitive operators (TensorFlow, 2017), executed in *execution frames* of the GPU that can be nested, allowing further optimizations. Thus, in the first parallel while loop of MB-MTCNN, the images are scaled and processed in batch using P-Net to obtain the region candidates. With a nested parallel while loop, the candidates of each scale and image are postprocessed following the original implementation; scaled to their real size, refined and filtered with a non-maximum suppression (NMS). Finally, the candidates from all the scales are grouped by image in batch and NMS is applied again to merge candidates of different scales using a parallel while loop. In the subsequent stages, the candidates are filtered and refined using R-Net and O-Net networks. In these stages, the batches of candidates from each image are processed in parallel. This is more efficient than processing all the candidates in one batch, as it avoids reallocating the candidates for preprocessing and postprocessing.

4.2 Efficient Dynamic Batching for Facial Images Processing

The number of cropped facial images that reach stages 2 and 3 of MSDB is unknown and can vary in every iteration. Different dynamic batching methods can be considered, which depend on the used GPU architecture and DNN deployment tool. We focus on NVIDIA GPUs and Google's TensorFlow and NVIDIA's TensorRT frameworks, due to their suitability for the inference of DNNs (Yadwadkar et al., 2019).

The naive dynamic batching approach consists in putting the whole batch of candidates directly into the network, varying the network's input size in every iteration. In TensorFlow constantly varying the batch size produces important time overheads, as the network needs to be re-adapted for the new size. TensorRT, optimizes the network for a given batch size, allowing handling smaller sizes with a drop in performance. Nevertheless, in both cases there are time overheads proportional to the network complexity.

Those time overheads due to network re-adaptation can be avoided using parallel loops, built upon flexible and expressive control-flow primitive operators (Agarwal, 2019; Radul et al., 2019). For instance, one can set the input size of the network to a certain value, so the original input batch is divided into mini-batches of that size, and process these mini-batches with TensorFlow's parallel while loop (TensorFlow, 2017). The mini-batch size must be set manually taking into account the expected minimum, maximum, and average batch sizes. Nevertheless, if there are big variations in the number of candidates, the number of mini-batches may be too high for a single parallelization, causing a drop in performance.

For this reason, our proposal for stages 2 and 3 of MSDB is to load and infer multiple instances of the network, optimized for different batch sizes. Thus, for each inference, we divide the input batch size in the minimum number of mini-batches that fit the different network sizes. In order to require less GPU memory to load and infer each network, we recommend applying optimizations to the trained network such as layer fusion, kernel auto-tuning, dynamic tensor memory and multi-stream execution.

5 EXPERIMENTAL RESULTS

In this section, we present the results of two groups of experiments conducted to evaluate the proposed clustering algorithm (ODIVC) and VSS. The first group aims to measure the performance of ODIVC in terms of accuracy and processing time, comparing it with other state-of-the-art offline clustering methods. The second group focuses on testing the performance and scalability of the proposed VSS and the impact of the optimizations presented in Section 4.

The server used for the experiments has the following specifications: 1 processor Intel XeonTM E5-1650v4, 1 GPU NVIDIA Quadro P6000 and 2 RAM modules, each one of 6GB DDR4 2400MHz.

5.1 Face Clustering Performance

For the first experiment, we select the IJB-B dataset (Whitelam et al., 2017), a well-known dataset of unconstrained in-the-wild face images. This dataset includes a clustering protocol consisting of seven subtasks that vary in the number of identities and the number of faces. We select the last subtask, as it is the most challenging one, with the highest number of identities (1,845) and faces (68,195). The performance is measured using BCubed F-Measure metric, following the recomendations in (Amigó et al., 2009).

Table 1: Comparison with baseline methods in terms of BCubed F-Measure and processing time using IJB-B-1845. Superscript* denotes results reported from original papers, otherwise it uses the i-vectors from (Wang et al., 2019).

Method	F-Meas	Time
ARO (Otto et al., 2018)	0.755	00:01:13
PAHC* (Lin et al., 2018)	0.61	00:03:56
ConPaC* (Shi et al., 2018)	0.634	02:53:58
DDC (Lin et al., 2018)	0.800	00:05:32
GCN (Wang et al., 2019)	**0.814**	00:06:03
ODIVC (ours)	0.778	**00:00:28**

Table 2: Comparison with baseline methods in terms of BCubed F-Measure and processing time using IJB-C dataset. All methods use the same i-vectors extracted from our VSS and the same hardware.

Method	F-Meas	Time
ARO (Otto et al., 2018)	0.768	00:09:39
GCN (Wang et al., 2019)	0.890	00:10:32
ODIVC (ours)	**0.931**	**00:00:52**

Since we want to demonstrate that ODIVC is independent of the recognition model used and a fair comparison with other methods, the same vectors used in (Wang et al., 2019) are selected for this experiment. These vectors have 512 dimensions. We adjust the parameters of ODIVC empirically: $thresh_{low}$ = 0.38, $thresh_{high}$ = 0.4 and nm_{min} = 4.

The results of the experiment are presented in Table 1. It can be observed that our method achieves the third position in terms of F-Measure, but outperforms the rest of the methods considering the processing time. For instance, it runs 12 times faster than GCN-A under the same hardware conditions.

In the second experiment we test the performance of ODIVC using a different face recognition model and a different dataset. We select the face recognition model used by our VSS, described in Section 3.2. We use IJB-C (Maze et al., 2018), another well-known dataset of unconstrained face images, with 3531 identities and 140623 faces. We use MB-MTCNN for the face and landmarks detection. In order to obtain better quality feature vectors, we filter faces with less than 45 pixels per side and we normalize the patches as described in Section 3.2. After filtering, 120661 vectors belonging to 3529 identities are extracted.

We compare our algorithm with the two best state-of-the-art methods: GCN (Wang et al., 2019), which achieved the highest accuracy in the first experiment,

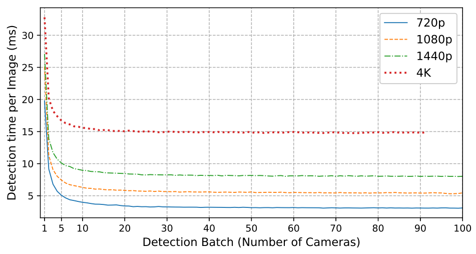


Figure 3: Detection time per image for different resolutions and batch sizes in MB-MTCNN (in stage 1 of MSDB), compared to MTCNN (batch=1).

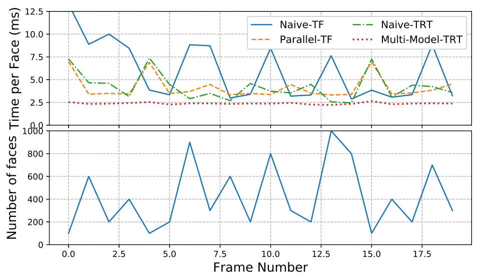


Figure 4: Average time per face of the dynamic batching procedure for stages 2 and 3 of MSDB, compared to alternative state-of-the-art approaches.

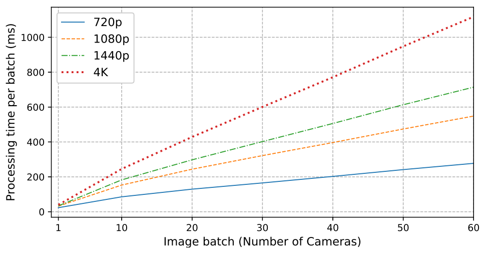


Figure 5: Average times per batch (for the image processing thread) with the considered setup for different resolutions.

and ARO (Otto et al., 2018), which achieved the best trade off between accuracy and speed (without considering ours). For both methods, we adjust the parameters in order to achieve the best performance. Furthermore, for GCN, we retrain the network following recommendations in (Wang et al., 2019). Finally, we tune the parameters of ODIVC: $thresh_{low}$ = 0.37, $thresh_{high}$ = 0.4 and nm_{min} = 4.

The results of this experiment, presented in Table 2, show that our method outperform the others in terms of F-Measure and processing time. Under the same hardware conditions ODIVC runs more than 12 times faster than GCN and more than 11 times faster than ARO. These time ratios show that our method is more scalable than the others. In this experiment ODIVC achieves a better F-Measure than GCN. We believe this is because we have reduced the number of outliers by filtering the smallest faces and using by a better face recognition network. Therefore, ODIVC is more sensible to outliers, but it is more accurate when using robust face representations.

5.2 Video Surveillance System Performance

We start evaluating the performance of MB-MTCNN compared to the original implementation. Figure 3 shows the average time of the detection stage per image for different resolutions and batch sizes using our approach (MTCNN corresponds to batch=1). The results show a great reduction in the processing time per image (more than 5 times for 720p). This reduction increases with the batch size but reaches a saturation point due to hardware limitations.

We also test the performance of the proposed dynamic batching procedure compared to the alternatives mentioned in Section 4.2. We measure the recognition time per face when the VSS is processing a sequence of images containing variable numbers of faces. To better visualize time variations, we augment the number of faces by a factor of 10, so they may vary from 10 to approximately 1000. The results are shown in Figure 4, where TF stands for TensorFlow and TRT for TensorRT. The mini-batch size selected for the Parallel-Loop-TF approach is 20 and those used for Multi-Instance-TRT are 400, 200, 100, and 20. It can be observed that the Multi-Instance-TRT outperforms the other approaches, not only in the average but also in the maximum peak times.

Finally, to test the potential scalability of our VSS, we run it to process images captured from a scaling number of videos, at different resolutions and with a detection batch size set with the same value as the number of videos. Then, we measure the average times per image batch in the image processing thread, the main bottleneck of the system. The results are shown in Figure 5. In our context, it is enough to deliver the PIs with near-real time performance. Hence, if we consider acceptable that the processing thread responds every 200 ms, this setup could theoretically be scaled up to 40 720p cameras. The results reveal that the proposed VSS allows designing cost-effective GPU-server-based solutions for our purpose.

6 CONCLUSIONS

In this work, we have presented a cost-effective VSS for face recognition and online clustering of unknown individuals at large scale, without storing their biometric information, in order to obtain PIs for people flow monitoring in large infrastructures. The VSS is composed of three main computing threads executed asynchronously, using CPU and/or GPU capabilities and sharing data sequentially. Experimental results with challenging scenarios reveal the high effectiveness and scalability of the proposed approach. Furthermore, we have presented an online and unsupervised clustering approach (ODIVC), which achieves state-of-the-art results in well-known large-scale datasets, with a reduced computational cost compared to the alternatives.

Future work will focus on extending our VSS to distributed computing infrastructures, with heterogeneous hardware as nodes of the VSS, including GPU computing servers that process and share data for video-surveillance purposes.

REFERENCES

Agarwal, A. (2019). Static automatic batching in TensorFlow. In *36th ICML*, volume 97, pages 92–101. PMLR.

Amigó, E., Gonzalo, J., Artiles, J., and Verdejo, M. (2009). Amigó e, gonzalo j, artiles j et ala comparison of extrinsic clustering evaluation metrics based on formal constraints. inform retriev 12:461-486. *Information Retrieval*, 12:461–486.

Chaudhari, S. T. and Kale, A. (2010). Face normalization: Enhancing face recognition. In *2010 3rd ICETET*, pages 520–525.

Chen, J., Li, K., Deng, Q., Li, K., and Yu, P. S. (2019). Distributed deep learning model for intelligent video surveillance systems with edge computing. *IEEE Trans. on Industrial Informatics*, pages 1–1.

Deng, J., Guo, J., Xue, N., and Zafeiriou, S. (2019). Arcface: Additive angular margin loss for deep face recognition. In *2019 IEEE CVPR*, pages 4685–4694.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE CVPR*, pages 770–778.

Jankar, J. R. and Shah, S. K. (2017). Computational analysis of hybrid high efficiency video encoders. In *ICISS*, pages 250–255.

Lim, K.-S., Lee, S.-H., Han, J. W., and Kim, G. W. (2018). Design considerations for an intelligent video surveillance system using cloud computing. In *PDCAT*, pages 84–89.

Lin, W., Chen, J., Castillo, C. D., and Chellappa, R. (2018). Deep density clustering of unconstrained faces. In *2018 IEEE/CVF CVPR*, pages 8128–8137.

Lloyd, S. P. (1982). Least squares quantization in pcm. *IEEE Trans. Inf. Theory*, 28:129–136.

Mayer, C. A., Felkel, R., and Peterson, K. (2015). Best practice on automated passenger flow measurement solutions. In *Journal of Airport Management*, volume 9, pages 144–153.

Maze, B., Adams, J., Duncan, J. A., Kalka, N., Miller, T., Otto, C., Jain, A. K., Niggel, W. T., Anderson, J., Cheney, J., and Grother, P. (2018). Iarpa janus benchmark - c: Face dataset and protocol. In *2018 ICB*, pages 158–165.

Nasir, M., Muhammad, K., Lloret, J., Kumar, A., and Sajjad, M. (2018). Fog computing enabled cost-effective distributed summarization of surveillance videos for smart cities. *Journal of Parallel and Distributed Computing*, 126.

Otto, C., Wang, D., and Jain, A. K. (2018). Clustering millions of faces by identity. *IEEE TPAMI*, 40(2):289–303.

Radul, A., Patton, B., Maclaurin, D., Hoffman, M. D., and Saurous, R. A. (2019). Automatically batching control-intensive programs for modern accelerators. *ArXiv*, abs/1910.11141.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE TPAMI*, 22:888–905.

Shi, Y., Otto, C., and Jain, A. K. (2018). Face clustering: Representation and pairwise constraints. *IEEE Transactions on Information Forensics and Security*, 13(7):1626–1640.

Tan, X. and Triggs, B. (2010). Enhanced local texture feature sets for face recognition under difficult lighting conditions. *IEEE Transactions on Image Processing*, 19(6):1635–1650.

TensorFlow, A. (2017). Implementation of control flow in tensorflow. *TensorFlow Whitepaper*.

Tsakanikas, V. and Dagiuklas, T. (2017). Video surveillance systems-current status and future trends. *Computers & Electrical Engineering*, 70.

Wang, H., Wang, Y., Zhou, Z., Ji, X., Gong, D., Zhou, J., Li, Z., and Liu, W. (2018). Cosface: Large margin cosine loss for deep face recognition. In *2018 IEEE/CVF CVPR*, pages 5265–5274.

Wang, Z., Zheng, L., Li, Y., and Wang, S. (2019). Linkage based face clustering via graph convolution network. *2019 IEEE/CVF CVPR*, pages 1117–1125.

Whitelam, C., Taborsky, E., Blanton, A., Maze, B., Adams, J., Miller, T., Kalka, N., Jain, A. K., Duncan, J. A., Allen, K., Cheney, J., and Grother, P. (2017). Iarpa janus benchmark-b face dataset. In *2017 IEEE CVPRW*, pages 592–600.

Yadwadkar, N. J., Romero, F., Li, Q., and Kozyrakis, C. (2019). A case for managed and model-less inference serving. In *HotOS '19*, pages 184–191.

Zhang, K., Zhang, Z., Li, Z., and Qiao, Y. (2016a). Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503.

Zhang, Z., Luo, P., Loy, C., and Tang, X. (2016b). Learning deep representation for face alignment with auxiliary attributes. *IEEE TPAMI*, 38(5):918–930.

4.5 Boosting Masked Face Recognition with Multi-Task ArcFace

- **Authors:** David Montero and Marcos Nieto and Peter Leskovský and Naiara Aginako
- **Booktitle:** Proceedings of the 16th International Conference on Signal Image Technology & Internet based Systems (SITIS)
- **Year:** 2010
- **Publisher:** IEEE

Boosting Masked Face Recognition with Multi-Task ArcFace

David Montero, Marcos Nieto, Peter Leskovsky and Naiara Aginako

***Abstract*— In this article, we tackle the recognition of faces wearing surgical masks. Surgical masks have become a necessary piece of daily apparel because of the COVID-19-related worldwide health problem. Modern face recognition models are in trouble because they were not made to function with masked faces. Furthermore, in order to stop the infection from spreading, apps capable of detecting if the individuals are wearing masks are also required. To address these issues, we present an end-to-end approach for training face recognition models based on the ArcFace architecture, including various changes to the backbone and loss computation. We also use data augmentation to generate a masked version of the original dataset and mix them on the fly while training. Without incurring any additional computational costs, we modify the chosen network to output also the likelihood of wearing a mask. Thus, the face recognition loss and the mask-usage loss are merged to create a new function known as Multi-Task ArcFace (MTArcFace). The conducted experiments demonstrate that our method outperforms the baseline model results when faces with masks are considered, while achieving similar metrics on the original dataset. In addition, it obtains a 99.78% of mean accuracy in mask-usage classification.**

I. Introduction

In recent years, advances in the field of face recognition have made it one of the most reliable biometric techniques among other existing techniques such as fingerprint recognition, hand geometry or iris scanning [1], [2]. Furthermore, compared to its alternatives, facial recognition has the following advantages:

- It is a more affordable solution than other alternatives such as fingerprint or iris scanners, as it only needs a mono camera as a sensor. In addition, several cameras may be connected to a single processing unit to even more reduce hardware costs.
- The verification can be done remotely; there is no need for the user to interact with the sensor.
- The sensor can be hidden, which can be very useful for security or aesthetic reasons.

All these features make face recognition the best choice for most of the applications based on human re-identification.

However, face recognition also has weaknesses. Cutting edge methods rely on Deep Neural Networks (DNN) that extract biometric feature vectors from the detected face images. These detected faces may have different orientations, lighting conditions, partial occlusions, low resolution, noise, etc., which can affect the robustness of the feature vectors [3].

David Montero, Marcos Nieto and Peter Leskovsky are with Vicomtech. Email: dmontero@vicomtech.org, mnieto@vicomtech.org, pleskovsky@vicomtech.org

David Montero and Naiara Aginako are with the Basque Country University. Email: naiara.aginako@ehu.eus

Most of these negative conditions can often be eliminated by selecting the correct hardware location and requirements and by preprocessing face images [4], but others such as partial occlusions caused by clothing accessories cannot be avoided. This specific problem has lately emerged as a significant barrier in the facial recognition field, especially since the global health crisis originated by COVID-19 has caused medical face masks to become an everyday-clothing-accessory.

The use of a mouth and nose-covering mask makes the face recognition models to lose about half of the useful biometric information. Since they have been designed to work with the whole face information, the quality of the feature vectors extracted from masked faces is compromised and the accuracy of the re-identification process decreases considerably, as stated in [5]. In fact, NIST agency recently released a study [6] where they analyzed the top commercial face recognition systems. For all the systems, they reported rates of mistakes from 5% to 50% in the re-identification of a subject wearing a mask.

While masked face detection has been widely studied and several robust solutions have been presented [7], [8], [9], masked face recognition remains an under-researched topic. In the last months, several masked face datasets [10], [11] and tools [12], [13] for generating synthetic data have been released. In addition, some methods trying to tackle this issue using different approaches have been presented [14], [13], [15]. Nevertheless, there is still much research to be done about this topic.

To contribute to this task, we propose an approach based on the ArcFace work presented by Deng et al. [16] with several modifications for the backbone and the loss function. From the original face-recognition dataset, we generate a masked version using data augmentation, and we combine both datasets during the training process. We modify the selected network, based on ResNet-50 [17], [18], to also output the probability that a face is wearing a mask without adding any additional computational cost. Furthermore, we combine the ArcFace loss with the mask-usage classification loss, resulting in a new function named Multi-Task ArcFace (MTArcFace).

Experimental results with non-masked and masked face-recognition validation datasets show that the proposed approach highly boosts the model accuracy when dealing with masked face recognition, while preserving almost the same accuracy on the non-masked datasets. Furthermore, the model achieves an accuracy of 99.78% in mask-usage classification.

II. Related Work

A. Face Recognition

Cutting edge methods for facial recognition rely on DNNs. They learn to identify the most relevant parts of the human faces an to represent them using feature vectors, maximizing the inter-class distance and minimizing the intra-class distance.

Most of these methods follow two training techniques. The first approach is based on a classification model where each class represents an identity, with a Softmax-based function for computing the loss [16], [19]. The other consists of learning the embeddings directly by minimizing the distance between the samples that belong to the same identity and maximizing the distance of the samples that belongs to different identities [20].

The accuracy for the two approaches is affected when dealing with facial masks [5], [6]. Nevertheless, as commented in [16], the models based on the second approach need an exhaustive analysis of the dataset before the training process to generate the groups of samples that will be compared. Considering this, we decided to follow the first approach to tackle the problem. In particular, we use ArcFace [16] work as our starting point, as it achieves the best accuracy among the state-of-the-art methods.

B. Masked Face Recognition

Numerous works have been released in an effort to resolve the masked face recognition challenge since the emergence of COVID-19. The suggested solutions attack the issue using various strategies that may be divided into three groups. The first one utilizes generative adversarial networks (GAN) to remove masks from the faces before passing them to the recognition model, avoiding the need of retraining it [21], [13]. Nevertheless, the reliability of the rebuilt faces, depends on the quality of the training process, the network, and the data. Furthermore, the time required for computation is considerably increased by this preprocessing step.

In the second group the models are trained using only the upper part of the face [15]. These models are faster, since they process a smaller region. However, they loose relevant information when working also with uncovered faces.

On the other hand, the third group address the challenge by combining faces with and without masks during the training process [12], [14]. For instance, the authors in [12] enrich the dataset VGG2 [22] with covered faces generated using data augmentation before training their model using the procedure presented in FaceNet [20]. A different approach is followed in [14], where the authors generate two embeddings per identity, corresponding to the masked and unmasked face images.

The method proposed in this work belongs to the third group, but using ArcFace [16] as the baseline model. First, we create a synthetic version with masks of a face recognition dataset using data augmentation. Then, during the training process, both datasets are shuffled separately using the same seed and, for every new face image selected for the input batch, we decide whether the image is taken from the original or the masked dataset with a probability of 50%. Furthermore, we take advantage of knowing to which dataset the face belongs to and modify the original network to output the probability that a face is wearing a mask without additional computational cost.

C. Masked Face Datasets

All these approaches require datasets of masked faces. To fulfill this need, several works have been presented. In [14], the authors release a dataset with identities containing masked and unmasked images with different orientations. Nevertheless the size of the dataset is reduced, with less than 12 thousand images and just one thousand identities, unsufficient for training a modern architecture like a ResNet-50 [17], [18]. Another dataset is released in [10], with 137 thousand masked faces. However, it is focused on detecting if a mask is correctly put on or not, and it does not provide ground truth for face recognition. In [11], the authors present two datasets, one real-world dataset with 95 thousand faces and 525 identities, and a synthetic dataset with 500 thousand faces 10 thousand identities. Nevertheless they are still not big enough for training a state-of-the-art face recognition model, compared for instance with the dataset used in ArcFace, MS1MV2, with almost 6M images and 85 thousand identities.

As an alternative, the authors in [12] propose a software for generating synthetic masks over real face images. It relies on a detector of facial landmarks to localize the region where the mask should be put. It provides several types of masks with different colors available. We take advantage of this software to create a masked version of our training dataset.

III. Proposed Method

A. Problem Definition

We consider the problem of facial recognition of subjects who may or may not wear masks. As we do not know if the subject is wearing a mask, the network must perform well in both cases. To solve this problem, we aim at increasing the accuracy of the face recognition network when dealing with masked faces, while preserving as much as possible the original accuracy with non-masked faces. In order to achieve this, the network must learn if the subject is wearing a mask to decide which facial features can be trusted in each case. We take advantage of this fact and modify the network so it also outputs the probability that the subject is wearing a mask.

B. Training Pipeline

Using the original dataset, we create a masked version of it and mix both on the fly while training. On every epoch, we shuffle each dataset with an identical seed and, with a probability of 50%, we randomly select if the incoming faces are taken from the masked or the original dataset. As mentioned in Section II-A, we use ArcFace [16] as the baseline work for two reasons: it uses a softmax-loss-based

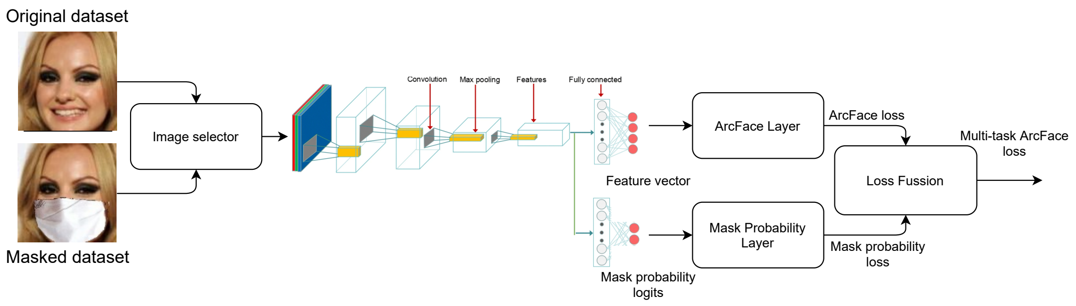


Fig. 1. Illustration of the proposed training pipeline. The image selector decides whether the next input image should be masked or not. The trained network is modified to output also the probability that the face is wearing a mask.

methodology, which does not require an exhaustive training-data-preparation stage; and it has been proven to be the approach that reports the best results for the original face recognition task. Thus, we select the dataset recommended in their work MS1MV2, a refinement of MS-Celeb-1M [23], composed of almost 6M images and 85 thousand identities. An illustration of the proposed training pipeline is shown in Figure 1.

We use the tool MaskTheFace [12] for creating the synthetic masked dataset. The types of masks considered are surgical, surgical green, surgical blue, N95, cloth and KN95. The type mask is selected randomly and there is a probability of 50% of applying a random color and a probability of 50% of applying a random texture. Some examples of the generated faces are shown in Figure 2.

We test several network architectures and, considering the balance between the parameter number and the accuracy, we choose LResNet-50. More specifically, we use our own implementation of the network in TensorFlow deep learning framework, publicly available in a GitHub repository [24].

Starting from this network, just after the dropout layer, we add a second linear layer, as shown in Figure 1. This layer outputs two logits representing the probabilities that the face is wearing a mask or not. By doing this, we compel the network to learn if a face is covered, knowledge that is also used by the face recognition head.

Thus, from the modified network we obtain two outputs, the logits (unnormalized predictions) of the ArcFace layer ($logits_{ArcFace}$) and the logits of the new dense layer ($logits_{Mask}$). To extract the combined error from both logits, we start by generating the ArcFace loss ($loss_{ArcFace}$) in the same way as in [24]:

$$loss_{ArcFace} = crossEnt(Softmax(logits_{ArcFace}, labels_{ID}) \tag{1}$$

Next, we calculate the loss associated with the probability of wearing a mask ($loss_{Mask}$) by applying the softmax activation function on the logits and cross-entropy with the labels:

$$loss_{Mask} = crossEnt(Softmax(logits_{Mask}), labels_{ID}) \tag{2}$$

The Multi-Task ArcFace loss ($loss_{MTArcFace}$) is obtained by adding these two losses. However, to reduce the impact of $loss_{Mask}$ and give more importance to the ArcFace loss, we use the logarithm of $loss_{Mask}$ instead of the original value:

$$loss_{MTArcFace} = loss_{ArcFace} + log(loss_{Mask} + 1.0) \tag{3}$$

Finally, we add the regularization loss (as in the original implementation) to compute the total loss that will be used for the optimization:

$$loss_{total} = loss_{MTArcFace} + loss_{regularization} \tag{4}$$

For training the model we use 2 GPUs Tesla V100, which allows us using 512 as the batch size. We train the model for 300k steps. We use the SGD optimizer with a momentum of 0.9 and an initial learning rate of 0.0015. The learning rate is reduced by a factor of 0.3 in steps 120k, 200k and 280k. The rest of the parameters of the network remain the same as in the original implementation. In Figure 3, we show the training loss curve and the face-recognition and mask-usage accuracy curves, compared to those of the original model.

IV. Experiments

In this section, we present the results of a series of experiments aimed at demonstrating the capabilities of the proposed method. We divide the experiments into two groups: identity verification and mask-usage verification.

A. Identity Verification

These experiments aim at measuring the improvement of the proposed method in the face verification task when dealing with masked faces. For measuring this increase, we use the original model as the baseline to compare the results.

For the verification task, we generate masked versions of 3 well-known face recognition datasets, also used in [16] for evaluating the original models:

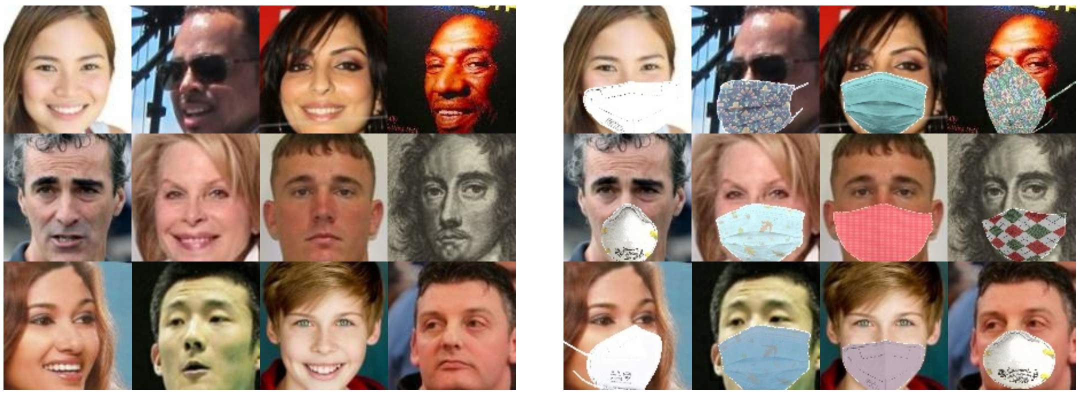


Fig. 2. Some examples of the training faces and their corresponding masked version generated with the MaskTheFace tool [12]

.

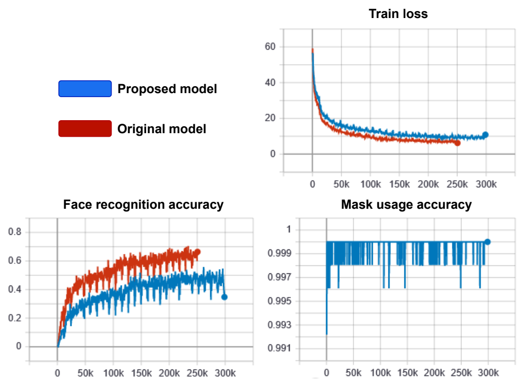


Fig. 3. Training curves for the proposed model and the original ArcFace model. The X axis represents the training steps

- Labeled Faces in the Wild (LFW) [25]: public benchmark for face verification containing 13,233 images from 5,749 people. The associated face verification task includes 6,000 comparisons.
- Celebrities in Frontal-Profile in the Wild (CFP) [26]: contains faces from 500 celebrities in frontal and profile views. Two verification protocols are presented for this dataset: one comparing only frontal faces (CFP_FF), and the other comparing frontal and profile faces (CFP_FP). Each of the protocols are composed of 7,000 comparisons. We consider both protocols for our experiments.
- Agedb [27]: the first manually collected, in-the-wild age database. Contains 16,488 images from 568 celebrities at different ages. Contains four verification protocols where the compared faces have an age difference of 5, 10, 20 and 30 years respectively. We select the last protocol for the experiments (AgeDB_30, as it is the most challenging one. It contains 6,000 comparisons.

TABLE I

Comparison of the verification performance (%) with the masked datasets between the proposed method and the original ArcFace model.

Dataset	Proposed Method	Original model
Masked LFW	98.92	94.75
Masked CFP_FF	98.33	92.73
Masked CFP_FP	88.43	76.81
Masked AGEDB_30	93.17	90.53
MFR2	99.41	97.17

In addition, we also consider for the experiment the masked face dataset MFR2 described in [12], with 269 real-world face images from 53 celebrities, where the 64% of the faces wear a mask. The associated verification process is composed of 848 comparisons. Some examples of the images of the different datasets considered for the experiment are shown in Figure 4.

We present the results of this experiment in Table I. They show that our method largely outperforms the original model in the face verification task when dealing with masked faces. This increase in performance is more evident with profile images, where the amount of information of the face available is reduced, as is the case with CFP_FP, where the proposed model is almost a 12% more accurate than the original.

We also want to test the accuracy of the new model when recognizing non-masked faces, to check whether it has been a significant drop of performance. Thus, we repeat the previous experiment with the original non-masked datasets and compare the results with those achieved by the original model. The results, exposed in Table II, show that there is indeed a drop of performance for the new model, but that it is not significant (less than a 2% in the worst case). Furthermore, this drop in performance is much less than the gain obtained with masked faces. For example, in the case

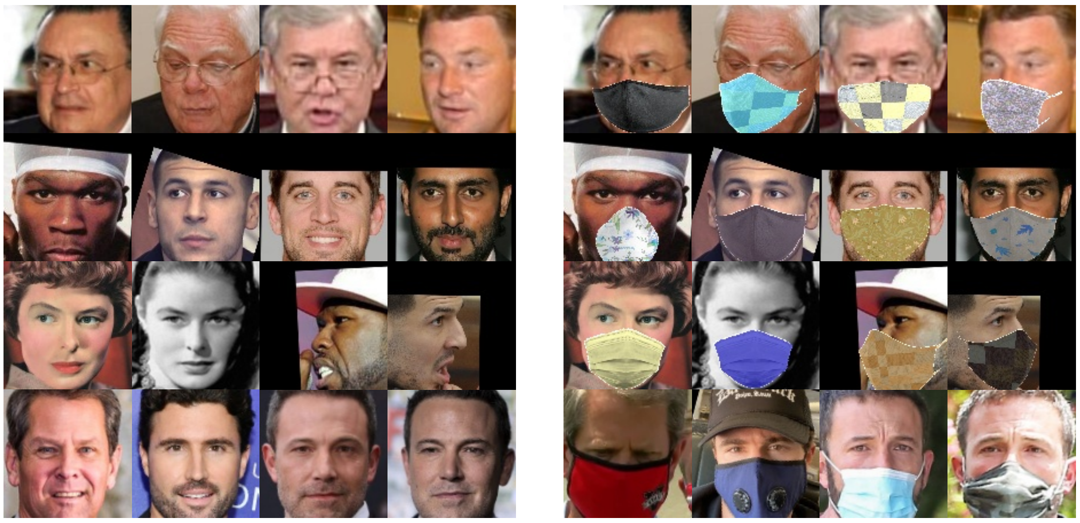


Fig. 4. Some examples of the faces of the evaluation datasets and their masked versions. The first row belongs to LFW [25], the second row to CFP [26], the third row to AGEDB [27] and the last row to MFR2 [12]
.

TABLE II
COMPARISON OF THE VERIFICATION PERFORMANCE (%) WITH THE ORIGINAL DATASETS BETWEEN THE PROPOSED METHOD AND THE ORIGINAL ARCFACE MODEL.

Dataset	Proposed Method	Original model
LFW	99.45	99.62
CFP_FF	99.40	99.70
CFP_FP	92.27	93.81
AGEDB_30	95.02	96.90

of CFP_FP, the model accuracy with masked faces increases almost a 12%, while its accuracy with non-masked faces decreases less than a 2%.

B. Mask-Usage Verification

Finally, we want to analyze the performance of the mask-usage probability output added to the proposed method. For this task, we run the model with all the faces contained in every masked and non-masked dataset used in the previous experiments. For each face we check whether the mask-usage probability estimated by the model is correct or not with a threshold of 0.5. Table I shows the results of the experiment. For each dataset, the model achieves nearly 100% accuracy. Again, the worst result is achieved for the CFP_FP dataset (98.82%) due to the profile faces. We believe that this is due to the fact that the training dataset does not contain enough profile faces. In any case, the model achieves an average accuracy of 99.78% across all datasets, so its effectiveness for this task is demonstrated.

TABLE III
MASK-USAGE VERIFICATION PERFORMANCE (%) OF THE PROPOSED METHOD.

Dataset	Accuracy
LFW	99.99
CFP_FF	99.99
CFP_FP	98.82
AGEDB_30	99.97
Masked LFW	99.98
Masked CFP_FF	99.98
Masked CFP_FP	99.70
Masked AGEDB_30	99.99

V. CONCLUSIONS

In this work, we have presented a full-training pipeline for ArcFace-based face-recognition models to adapt them for working with masked faces. This pipeline includes the generation of a synthetic masked dataset from the original training dataset. Furthermore, we have taken advantage of knowing to which dataset the face belongs to and modified the original network to output the probability that a face is wearing a mask without additional computational cost. As a result, we have created a new loss function to teach the network to extract vectors of good quality and reliable mask-usage probabilities called Multi-Task ArcFace. Experimental results with multiple masked and non-masked datasets have demonstrated that the proposed method highly boosts the performance of the model when recognizing masked faces, while suffering just a small drop in performance with non-masked faces. Furthermore, it has also been demonstrated its effectiveness for the mask-usage verification task with an average performance of 99.78% of accuracy across all datasets.

Future work will focus on extending the applicability of this method to other types of occlusions, such as eyes-masked faces. In addition, we will also study the possibility of adding a new output to the model to classify if the subject is wearing a mask correctly or if it is wearing it under its nose or its mouth.

VI. Acknowledgements

This paper is supported by European Union's Horizon 2020 research and innovation programme under grant agreement No 883341, project GRACE (Global Response Against Child Exploitation).

References

[1] D. Mou, *Fundamentals and Advances in Biometrics and Face Recognition*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, ch. 2, pp. 13–70.

[2] N. Anot and K. Singh, "A review on biometrics and face recognition techniques." *International Journal of Advanced Research*, vol. 4, pp. 783–786, 05 2016.

[3] X. Tan and B. Triggs, "Enhanced local texture feature sets for face recognition under difficult lighting conditions," *IEEE Transactions on Image Processing*, vol. 19, no. 6, pp. 1635–1650, June 2010.

[4] S. T. Chaudhari and A. Kale, "Face normalization: Enhancing face recognition," in *2010 3rd International Conference on Emerging Trends in Engineering and Technology*, Nov 2010, pp. 520–525.

[5] N. Damer, J. H. Grebe, C. Chen, F. Boutros, F. Kirchbuchner, and A. Kuijper, "The effect of wearing a mask on face recognition performance: an exploratory study," in *2020 International Conference of the Biometrics Special Interest Group (BIOSIG)*, 2020, pp. 1–6.

[6] "Nist finds flaws in facial checks on people with covid masks," *Biometric Technology Today*, vol. 2020, no. 8, p. 2, 2020.

[7] M. Jiang, X. Fan, and H. Yan, "Retinamask: A face mask detector," 2020.

[8] M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, "A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the covid-19 pandemic," *Measurement*, vol. 167, p. 108288, 2021.

[9] S. Lin, L. Cai, X. Lin, and R. Ji, "Masked face detection via a modified lenet," *Neurocomputing*, vol. 218, pp. 197 – 202, 2016.

[10] A. Cabani, K. Hammoudi, H. Benhabiles, and M. Melkemi, "Maskedface-net - a dataset of correctly/incorrectly masked face images in the context of covid-19," *ArXiv*, vol. abs/2008.08016, 2020.

[11] Z. Wang, G. Wang, B. Huang, Z. Xiong, Q. Hong, H. Wu, P. Yi, K. Jiang, N. Wang, Y. Pei, H. Chen, Y. Miao, Z. Huang, and J. Liang, "Masked face recognition dataset and application," 2020.

[12] A. Anwar and A. Raychowdhury, "Masked face recognition for secure authentication," *ArXiv*, vol. abs/2008.11104, 2020.

[13] N. Ud Din, K. Javed, S. Bae, and J. Yi, "A novel gan-based network for unmasking of masked face," *IEEE Access*, vol. 8, pp. 44 276–44 287, 2020.

[14] M. Geng, P. Peng, Y. Huang, and Y. Tian, "Masked face recognition with generative data augmentation and domain constrained ranking," in *Proceedings of the 28th ACM International Conference on Multimedia*, ser. MM '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 2246–2254.

[15] W. Hariri, "Efficient masked face recognition method during the covid-19 pandemic," 2020.

[16] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[17] D. Han, J. Kim, and J. Kim, "Deep pyramidal residual networks," in *2017 CVPR*, 2017, pp. 6307–6315.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 CVPR*, June 2016, pp. 770–778.

[19] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *2017 CVPR*, 2017, pp. 6738–6746.

[20] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the CVPR*, June 2015.

[21] C. Li, S. Ge, D. Zhang, and J. Li, "Look through masks: Towards masked face recognition with de-occlusion distillation," in *Proceedings of the 28th ACM International Conference on Multimedia*, ser. MM '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 3016–3024.

[22] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "Vggface2: A dataset for recognising faces across pose and age," 2018.

[23] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao, "Ms-celeb-1m: A dataset and benchmark for large-scale face recognition," in *ECCV*, vol. 9907, 10 2016, pp. 87–102.

[24] D. Montero, "face_recognition_tf2," 2019. [Online]. Available: https://github.com/dmonterom/face_recognition_TF2

[25] G. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments," *Tech. rep.*, 10 2008.

[26] S. Sengupta, J. Cheng, C. Castillo, V. Patel, R. Chellappa, and D. Jacobs, "Frontal to profile face verification in the wild," in *IEEE Conference on Applications of Computer Vision*, February 2016.

[27] S. Moschoglou, A. Papaioannou, C. Sagonas, J. Deng, I. Kotsia, and S. Zafeiriou, "Agedb: the first manually collected, in-the-wild age database," in *Proceedings of the CVPR Workshop*, vol. 2, 2017, p. 5.

4.6 BEV Object Tracking for LIDAR-based Ground Truth Generation

- **Authors:** David Montero and Nerea Aranjuelo and Orti Senderos and Marcos Nieto
- **Booktitle:** Proceedings of the 2019 27th European Signal Processing Conference (EUSIPCO)
- **Year:** 2019
- **Publisher:** IEEE

BEV Object Tracking for LIDAR-based Ground Truth Generation

David Montero*, Nerea Aranjuelo*, Orti Senderos* and Marcos Nieto*
*Vicomtech, Mikeletegi 57, P. Tecnológico, 20009, San Sebastián, Spain
Email: see http://www.vicomtech.org

***Abstract*—Building ADAS (Advanced Driver Assistance Systems) or AD (Autonomous Driving) vehicles implies the acquisition of large volumes of data and a costly annotation process to create labeled metadata. Labels are then used for either ground truth composition (for test and validation of algorithms) or to set-up training datasets for machine learning processes. In this paper we present a 3D object tracking mechanism that operates on detections from point cloud sequences. It works in two steps: first an online phase which runs a Branch and Bound algorithm (BBA) to solve the association between detections and tracks, and a second filtering step which adds the required temporal smoothness. Results on KITTI dataset show the produced tracks are accurate and robust against noisy and missing detections, as produced by state-of-the-art deep learning detectors.**

I. Introduction

In recent years, huge improvements have been made in the field of Advanced Driver Assistance Systems (ADAS). Current ADAS are capable to perceive the surrounding environment of the vehicle by using sensors and take real-time decisions for safety (emergency braking, lane departure warning) or comfort (lane keeping system, automatic cruise control).

Most of these achievements have been possible thanks to the continuous and rapid advances in Deep Learning (DL) [1][2]. New DL models can process the data captured by different sensors, such as cameras or LIDARs, to obtain a precise estimation of the scene (e.g. other vehicles, pedestrians, traffic signs). However, the accuracy of these models depends almost entirely on the richness of the dataset used for training.

DL models require training with huge amounts of data precisely annotated. The annotation process is costly and slow, usually requiring large groups of human operators creating the labels with specific tools. This process is the main bottleneck for the improvement of DL and as a consequence, ADAS.

In order to reduce this cost and accelerate the process, many annotation tools, automatic and semi-automatic, have been developed using different approaches [3][4]. Though, automatic annotation tools are not perfect and they introduce errors or inaccuracies to the annotations, which need always to be corrected or, in the best case, validated as ground truth. In the case of object annotation, it is critical that the automated step produce labels which are not only accurate in space (e.g. a bounding box in 2D or cuboid in 3D), but also coherent in time, thus assigning single identifiers to objects through the sequence. In this paper we propose an offline 3D object tracking algorithm as a part of a semi-automatic annotation tool for LIDAR data.

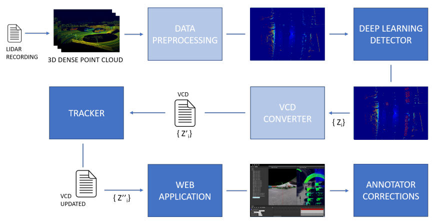


Fig. 1. Diagram of the annotation process.

Multi-object tracking is usually focused on solving how to associate incoming new detections to existing tracks [5], in an online process whose main requirement is to operate real-time. Offline processes, on the contrary, tend to find a graph with all possible associations and solve it in a joint optimized process, which result in very slow batch processes.

In our work we propose an approach which is as fast as online processes, by means of defining an online frame-level association problem, but solving it with a recursive function which ensures optimal association, plus additional post-processing steps which provide the necessary estimation smoothness. Our approach aims to obtain the highest possible accuracy in the least possible time in order to work effectively in a semi-automatic annotation process.

II. System overview

Figure 1 illustrates the pipeline of the annotation tool which includes the proposed tracking component. The annotation process is divided in a series of steps, starting with the generation of the recordings from the sensorized vehicles.

LIDAR streams (3D point cloud sequences) are then pre-processed to create bird's-eye view (BEV) images (also called top view images, see section III-A), which are then used as input for the Convolutional Neural Network (CNN) detectors (see section III). The detector outputs are cuboids in 3D space, defined as $\mathbf{Z}_{t,n} = (x, y, z, r_x, r_y, r_z, w, h, l, c, s)$, where t is the time step, n is the detected object number in that time step, $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ are the object center coordinates, $\mathbf{r_x}$, $\mathbf{r_y}$ and $\mathbf{r_z}$ are the object rotation angles, $\mathbf{w}$, $\mathbf{h}$ and $\mathbf{l}$ are it width, height, and length, $\mathbf{c}$ is the class number, and $\mathbf{s}$ is the confidence of the detection. This data will be converted into a standard format using the VCD converter[1], and will serve as the input

[1]https://vicomtech.box.com/v/vcd-library-linux-windows

data for the tracker. The tracker will associate the input object detections between the different time steps, generate predictions where detections are missing and finally correct the input object properties by applying a post-process to the generated tracks (see section IV). As a result, the VCD payload describing the scene is updated with the tracking information and sent to a web application for the final annotation step carried out by human operators.

III. Object Detection

Using latest advances in object detection with DL algorithms [1], we have trained a CNN to generate oriented bounding box detections based on LIDAR point clouds.

A. 3D Point Cloud Representation

A Velodyne HDL-64E laser scanner produces about one million 3D points per second [6]. Applying detection algorithms directly on these point clouds is normally avoided because of their sparse nature and the large amount of data to be processed [7][8]. Due to the high dimensionality and sparsity of the data, we have adopted an approach based on BEV map data representation. The point cloud is projected to the ground plane and discretized into a 2D grid of cells with resolution of 0.1 x 0.1m. Only the points inside a range of [(-40, 40), (-40, 40), (-1.75, 1.25)] meters are considered, taking the LIDAR position as the origin of coordinates. For each cell, the minimum, average and maximum heights of all the contained points are stored. These 3 features are stored as a 3-channel matrix with size of 800×800 and fed into a CNN.

B. Deep neural network

The trained network takes the encoded feature maps as input and produces oriented 3D boxes for the considered object classes. The network is based on Faster R-CNN architecture [9], which consists on two main stages. First, a Region Proposal Network (RPN) generates bounding boxes with object candidates. In the second stage, for each box proposal, extracted features are used to classify it, as well as to regress the final box coordinates as well as its rotation angle. Non-maximum suppression (NMS) is used to reduce redundancy of highly overlapped boxes. We use a IoU (Intersection-over-Union) threshold of 0.7 for NMS. The backbone network we use for feature extraction is ResNet-101 [10].

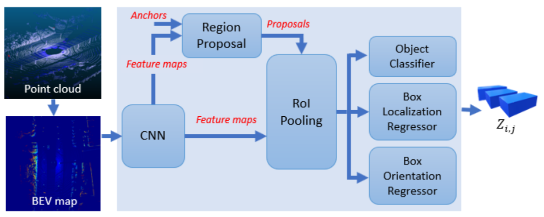


Fig. 2. Neural network architecture for oriented box detection.

The model is optimized for a multi-task loss function, which combines classification and bounding box regression losses [9]. Different from the original Faster R-CNN architecture, which does not predict the orientation of the boxes, an extra loss term is added [11] to minimize the error between the estimated rotated box and the ground truth.

The network is trained in an end-to-end fashion. Weights are initialized with pretrained weights on ImageNet data [12]. The training dataset is generated with the sequences published so far from the public driving dataset nuScenes [13]. Augmentation techniques are applied to augment samples containing the least frequent classes. The total amount of training samples is 20000, including 9 different object classes: car, pedestrian, cyclist, train, truck, bus, motorist, construction vehicle and trailer. Custom anchor boxes are designed for each class.

For each time step t, each object prediction is parameterized by $\mathbf{Z}_{t,d}$ as defined in previous section. The height of the bounding box is estimated based on the object class.

IV. Tracking

As mentioned in section II, the tracker is in charge of associating the input objects between the different time steps, generating missing tracking states in intermediate steps and correcting the input object properties by applying a post-process to the generated tracks. A diagram describing the tracking process can be found in Figure 3.

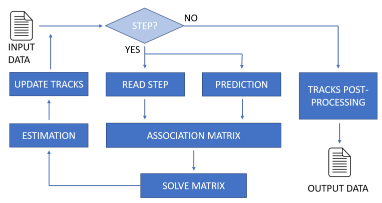


Fig. 3. Offline tracking dataflow where input data stands for the obtained CNN detections, and output data are the updated tracks.

The tracking process is divided into two main components. First, the online component is executed as a loop where, at each iteration, a new time step (i.e. frame) of the input data is processed. Tracks are then updated using predictions based on their previous information and the new input detections. In the correction step an association matrix is generated and solved in order to match existing tracks predictions with detected objects and the tracks are updated. Finally, a post-process adds smoothness to produced tracks both in position and rotation.

A. Linear prediction

It is assumed that detections are filtered by their score using a suitable user-defined threshold before feeding the tracker, so the variable of $\mathbf{Z}_{t,n}$ is not used during the tracking process.

After analyzing the data and the use case, the following conclusions were reached. z position, r_x and r_y is approximately constant in almost every case, therefore it does not provide relevant information. Also, r_z is a noisy variable and will be corrected in the post-process. So we decided to discard these variables from the prediction model. Similarly, the object size and the class variables are assumed to be constants, so they are not considered for the prediction.

After this filter, the prediction variables are x, y and r_z. For solving this problem, it was decided to use a constant acceleration model. So, considering all the previous information, a track state will be represented by $\mathbf{S}_{t,m} = (x, y, \dot{x}, \dot{y}, \ddot{x}, \ddot{y}, z, r_x, r_y, r_z, w, h, l, c)$, where t is the time step, and m is the index of the track.

B. Association matrix

For the association between the tracks states and the detections an association matrix is defined. Let this matrix be $A_{m \times n}$ where M is the number of existing tracks, and N is the number of incoming detections.

Each entry of A encodes the association likelihood (between 0 and 1) of track m and detection n. Before computing the likelihood, it is checked that the track and the detection class belongs to the same group. Two groups are defined based on the detector output classes: a vehicle class (cars, trucks and buses) and a human class (pedestrians, cyclists and bikers). This is applied because classes of the same group are more likely to be confused by the detector.

Only the 2D position variables (x and y) are taken in account for the likelihood computing, due to the noise in the r_z, s_x and s_y. The likelihood function can be selected as any decaying function around the predicted state of the track. Therefore, we need first to define the distance between the centroids of the predicted track $\mathbf{p} = (p_x, p_y)^\top$ and the detection $\mathbf{q} = (q_x, q_y)^\top$, as $\mathbf{d'} = (\mathbf{p} - \mathbf{q})$.

The model must include the uncertainty of the prediction itself and the noise of the detections. For convenience, we use a bivariate normal distribution on a normalized distance $\mathbf{d}$ between the centroids of the predicted track and the detection. This distance will be rotated so the $\mathbf{x}$ axis aligned with the velocity vector, using $\theta = atan(v_y, v_x)$:

$$d' = d' \begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix} \tag{1}$$

Thus, the distance is normalized as:

$$\mathbf{d} = \left(\frac{max(0, |d'_x| - \epsilon)}{d_{max}}, \frac{max(0, |d'_y| - \epsilon)}{d_{max}} \right)^\top \tag{2}$$

where ϵ is the detection noise, and d_{max} is the maximum association distance, which depends on the class group of the track. This normalization ensures that the distance measure includes the detection noise.

The likelihood function is then defined as:

$$\mathcal{L}(\mathbf{d}) = \frac{1}{\sqrt{(2\pi)^2 |\sigma|}} exp(-\frac{1}{2}(\mathbf{d}^\top \sigma^{-1} \mathbf{d})) \tag{3}$$

where σ is the covariance matrix which defines the shape of the likelihood function. As we need a likelihood function ranged between 0 and 1, we need to select $\sigma_x = \sigma_y = \frac{1}{\sqrt{2\pi}}$. Then, the likelihood is redefined as:

$$\mathcal{L}(\mathbf{d}) = exp(-(||\mathbf{d}||^2 \pi)) = exp(-d_x^2 \pi) exp(-d_y^2 \pi) \tag{4}$$

As we want to include the uncertainty of the prediction process, we truncate this function to a certain limit, defined by an ellipse aligned with the velocity vector of the track and which axis length will depend on the prediction certainty. The axes of the ellipse can be defined as:

$$\begin{aligned} s_x &= d_{max} + (1 + ||v||(\Delta t + 1))\gamma_x \\ s_y &= d_{max} + (1 + ||v||(\Delta t + 1))\gamma_y \end{aligned} \tag{5}$$

where $||\mathbf{v}||$ is the magnitude of the velocity vector of the track, Δt depends on the number of steps passed since the last association of a detection with the track, and γ_x and γ_y are factors which elongate the ellipse according to the expected object dynamics (e.g. for cars, we have found that $\gamma_x = 2.0$ and $\gamma_y = 0.3$ encode well its inertia).

Then, the ellipse is defined as $\mathbf{d}S^{-1}\mathbf{d} = 1$, where S is the matrix that defines the axes of the ellipse: $S = diag(s_x^2, s_y^2)$.

C. Association matrix resolution

Solving correctly the association matrix is crucial for the tracking, as a wrong or sub-optimal match in a time step propagates the error for the rest of the sequence. Two ways of solving it are considered and tested: (i) the Greedy Algorithm (GA)[14], used as baseline for comparison; and (ii) our proposed modified Branch and Bound Algorithm (BBA) [15].

GA works selecting individually the global maximum out of the matrix and setting to zero its corresponding row and column, and iterating finding next maxima until the matrix is emptied. BBA works, instead, in a recursive manner, finding maxima and then exploring the tree of possible suboptimal associations in order to find better joint likelihoods for the entire matrix. BAA guarantees that the sum of the likelihood of the produced associations is optimal, while GA does not.

GA is usually faster than BBA when there is a high number of candidates, but results using BBA are better in most of the cases. The two algorithms were implemented and tested in different scenarios. The results of some of those tests are presented in table I, where the errors are the number of steps in which the sum of the likelihood is worse using the GA, Max diff is the maximum difference of the BBA and GA likelihood sum registered, GA time is the average time of the GA implementation execution and the BBA time is the algorithm implementation execution, both in microseconds and per frame. Although this table shows relatively small errors of the GA results at frame-level, the impact of these errors is extremely significant as they are propagated to the rest of the sequence causing erroneous tracks.

TABLE I
ASSOCIATION ALGORITHMS COMPARISON

Scenario	Steps	Errors	Max diff	GA time	BBA time
Street road	611	10	0.423	20 μs	35 μs
Parking 1	189	5	0.703	29 μs	45 μs
Parking 2	404	6	0.559	110 μs	132 μs

Considering the tests results and the priority of the accuracy over the processing time it was decided to use a modified

BBA as the matrix solver. In the proposed implementation, the algorithm starts associating each track with the detection with the best likelihood. Then it looks for collisions (where a collision stands for the case where two tracks have been associated with the same detection). If it finds any, then a recursive function will be called until it finds the best collisions free combination. The algorithm can be found in algorithm 1, where candidates variable is the association vector.

Algorithm 1 B&B Recursive Function

```
procedure SOLVECOL($candidates, a, b$)
    $candidatesA \leftarrow candidates$
    $candidatesA(a) \leftarrow getNextCandidate(a)$
    $candidatesA \leftarrow checkCol(candidatesA)$
    $scoreA \leftarrow \sum_{i=0}^{n\_tracks-1} \mathcal{L}_{i,candidatesA(i)}$
    $candidatesB \leftarrow candidates$
    $candidatesB(b) \leftarrow getNextCandidate(b)$
    $candidatesB \leftarrow checkCol(candidatesB)$
    $scoreB \leftarrow \sum_{i=0}^{n\_tracks-1} \mathcal{L}_{i,candidatesB(i)}$
    if $scoreA \geq scoreB$ then
        return $candidatesA$
    return $candidatesB$
procedure CHECKCOL($candidates$)
    for $i \leftarrow 0; i < n\_tracks - 1; i \leftarrow i + 1$ do
        for $j \leftarrow i + 1; j < n\_tracks; j \leftarrow j + 1$ do
            if $candidates(i) = candidates(j)$ then
                $candidates \leftarrow solveCol(candidates, i, j)$
    return $candidates$
```

D. Estimation and tracks updating

Once the association matrix is solved, the tracks new state will be estimated. The position estimation will be calculated correcting the detection position using the equation 6, where $\mathbf{p}$ can be x or y, $dist_p$ is the difference between prediction and detection positions, $\mathbf{predn}_p$ is the prediction noise, which depends on the velocity, the number of steps without detections (Δt) and a noise coefficient, and $\mathbf{detn}$ is the detection noise.

$$p_{est} = p_{det} + (max(0, dist_p - max(predn_p - detn), 0)))/2$$
$$predn_p = (dist_p/||dist_p||)||v_p||\Delta t \gamma_p \tag{6}$$

The velocity in $\mathbf{x}$ and $\mathbf{y}$ is calculated by the difference between the new and the last estimated positions divided by the difference of steps, and smoothed using a linear interpolation with the last 3 velocity values registered. The same method is used for the acceleration in $\mathbf{x}$ and $\mathbf{y}$. For the rest of the state variables, the value of the detection variables will be assign, since they will be treated in post-processing.

After the new tracks states are computed, the missing tracks states from previous steps will be generated using a linear interpolation. Also, it will be checked if there are dead tracks (without an associated detection in the last $\mathbf{n}$ steps) and separate them from the active ones. Finally, new tracks will be generated using the unassociated detections.

E. Tracks post-processing

Once the online stage has finished, the post-processing stage will begin. The aim of this stage is to remove orphan tracks and to correct the detection noisiest variables, the z axis rotation angle and the class. They are considered as orphan tracks all the tracks that has less than $\mathbf{n}$ states, and they will be removed, since they have a high probability of being erroneous.

For the $\mathbf{r}_z$ correction, it is assumed that, in most cases, the detection value is right, and that it wont change drastically from one step to another, so it will be calculated using the mode between $\mathbf{s} - \mathbf{n}$ and $\mathbf{s} + \mathbf{n}$ steps, being $\mathbf{s}$ the current step number. Also, the track size will be corrected using the mode of every track states, as it should be constant in every step. Finally, for the track class variable, it is again assumed that, in most cases, the detection value is right, so it will receive the value of the mode between all track steps.

V. Tests & discussion

In this section we present an experimental analysis of our method. We validate the tracker using the first 10 of the 22 training sequences with ground truth of the KITTI dataset with the tracking metrics proposed in [16].

In the ideal situation when detections are perfect, using the ground truth boxes from the KITTI dataset, we have observed that the tracker produces perfect tracks, without error.

A. Ablation study

We perform an ablation study on the input data in order to measure the impact of imperfect detections on the tracking results. We define three types of detection problems: (i) spatial noise; (ii) temporal sparsity; and (iii) a combination of both. The first test evaluates the effect of translation, rotation and size measurement noise in the detections. The temporal sparsity refers refers to the miss-detections of objects in specific frames. For the first test define a probability of 50% to apply randomly a noise between -20% and 20% to each detection in each frame. The experiment is repeated 10 times; the average results for each sequence are shown in Table II.

TABLE II
Tracker results with spatial noise

Seq	MOTA	F1	MT	PT	ML	FRAG
0	0.8626	0.9319	12.0	0.0	0.0	33.0
1	0.8561	0.9292	90.4	1.6	0.0	177.1
2	0.8684	0.9346	15.6	0.4	0.0	68.4
3	0.8485	0.9253	9.0	0.0	0.0	25.8
4	0.8511	0.9270	29.3	1.6	0.1	59.7
5	0.8402	0.9212	33.5	0.5	0.0	91.6
6	0.8383	0.9203	12.8	0.2	0.0	48
7	0.8520	0.9268	56.7	0.3	0.0	163.2
8	0.8459	0.9237	24.2	0.8	0.0	92.8
9	0.8664	0.9338	87.0	1.0	0.0	185.9

Temporal sparsity is analyzed following the same process and suppressing 20% of each object detections randomly across the sequence (see Table III).

For the third test we combine the spatial noise and the temporal sparsity to simulate a real detector (see Table IV). We can observe the tracker is able to solve temporal sparsity better than spatial noise, since, in the correction stage, the tracker assumes the position of detections are highly reliable.

TABLE III
TRACKER RESULTS WITH TEMPORAL NOISE

Seq	MOTA	F1	MT	PT	ML	FRAG
0	0.9897	0.9964	12.0	0.0	0.0	3.8
1	0.9940	0.9983	92.0	0.0	0.0	8.4
2	0.9984	0.9996	16.0	0.0	0.0	0.9
3	0.9907	0.9977	9.0	0.0	0.0	1.8
4	0.9987	0.9997	31.0	0.0	0.0	0.6
5	0.9985	0.9995	34.0	0.0	0.0	1.0
6	0.9991	0.9998	13.0	0.0	0.0	0.3
7	0.9981	0.9995	57.0	0.0	0.0	2.4
8	0.9997	0.9999	25.0	0.0	0.0	0.2
9	0.9913	0.9977	88.0	0.0	0.0	13.4

TABLE IV
TRACKER RESULTS WITH COMBINED NOISE

Seq	MOTA	F1	MT	PT	ML	FRAG
0	0.8770	0.9409	12.0	0.0	0.0	29.6
1	0.8778	0.9417	91.0	1.0	0.0	146.5
2	0.8935	0.9472	15.9	0.1	0.0	55.3
3	0.8454	0.9251	9.0	0.0	0.0	28.0
4	0.8735	0.9380	29.9	2.0	0.1	47.3
5	0.8640	0.9340	33.4	0.6	0.0	73.6
6	0.8693	0.9360	13.0	0.0	0.0	35.3
7	0.8689	0.9360	56.9	0.1	0.0	144.8
8	0.8771	0.9392	24.2	0.8	0.0	73.3
9	0.8783	0.9419	87.5	0.5	0.0	166.7

B. Study on generated detections

We finally analyze the results of the tracking when the input estimations are generated by the trained point cloud based object detector (see section III). The same evaluation metrics are computed for this experiment. Results are shown in Table V, where there is an extra column (F1D) with the F-Score obtained with the detector output. In this experiment the MOTA results with real detections are, as expected, worse than with ground truth detections due to their inherent noise. However, there is always an improvement of F-Score, ranging between $2-5\%$ thanks to the usage of the tracker.

TABLE V
TRACKER RESULTS WITH REAL DETECTOR

Seq	MOTA	F1	F1D	MT	PT	ML	FRAG
0	0.5476	0.7650	0.7468	5	6	1	14
1	0.5389	0.7631	0.7420	50	20	21	92
2	0.3094	0.6491	0.6340	6	5	0	20
3	0.6265	0.8095	0.7800	4	2	3	9
4	0.6814	0.8490	0.7965	20	9	1	25
5	0.6280	0.8150	0.7077	14	9	11	34
6	0.7438	0.8788	0.8382	8	3	2	15
7	0.7000	0.8542	0.8473	43	11	3	70
8	0.5069	0.7554	0.7290	12	8	4	26
9	0.3406	0.6487	0.6382	29	26	27	106

VI. CONCLUSIONS

In this work, we have shown our implementation of a 3D object offline tracking technique, which is robust against noisy and sparse detections produced by deep learning detection frameworks. Our approach combines the benefits of online tracking schemas, and thus operating near real-time, and the reliability of batch processes, by means of applying a recursive implementation of the Branch and Bound algorithm (BBA) to optimally solve the association problem.

In our experiments we show the BBA algorithm is optimal with respect the usual Greedy Algorithm (GA) approach, while keeping its computation under real-time requirements. Additionally, we have explored the impact of the noise and sparsity of detections benchmarking our proposed work against the ground truth from the KITTI dataset.

Furthermore, the proposed tracker is integrated into a web-based annotation platform which takes the produced tracks and presents them to teams of human annotators which refine, correct and finally validate the annotations. Future work will include a refinement of the score function at track level, so that the interaction between the algorithm and the annotators can be extended to offer finer level of control to the users.

VII. ACKNOWLEDGMENTS

This work has received funding from the European Commission (EC) Horizon 2020 programme (grant agreement no. 688099, project Cloud-LSVA).

REFERENCES

[1] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *arXiv preprint arXiv:1809.02165*, 2018.

[2] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, "A review on deep learning techniques applied to semantic segmentation," *arXiv preprint arXiv:1704.06857*, 2017.

[3] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: a database and web-based tool for image annotation," *International journal of computer vision*, vol. 77, no. 1-3, pp. 157–173, 2008.

[4] S. Bianco, G. Ciocca, P. Napoletano, and R. Schettini, "An interactive tool for manual, semi-automatic and automatic video annotation," *Computer Vision and Image Understanding*, vol. 131, pp. 88–99, 2015.

[5] H. S. Parekh, D. G. Thakore, and U. K. Jaliya, "A survey on object detection and tracking methods," *IJIRCCE*, vol. 2, pp. 2970–2979, 2014.

[6] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361, IEEE, 2012.

[7] Y. Zeng, Y. Hu, S. Liu, J. Ye, Y. Han, X. Li, and N. Sun, "Rt3d: Real-time 3-d vehicle detection in lidar point cloud for autonomous driving," *IEEE Robotics and Automation Letters*, vol. 3, pp. 3434–3440, 2018.

[8] B. Yang, W. Luo, and R. Urtasun, "Pixor: Real-time 3d object detection from point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7652–7660, 2018.

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[11] Y. Jiang, X. Zhu, X. Wang, S. Yang, W. Li, H. Wang, P. Fu, and Z. Luo, "R2cnn: rotational region cnn for orientation robust scene text detection," *arXiv preprint arXiv:1706.09579*, 2017.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[13] "Nuscenes." https://www.nuscenes.org/. (Accessed on 25/02/2019).

[14] J. Edmonds, "Matroids and the greedy algorithm," *Mathematical programming*, vol. 1, no. 1, pp. 127–136, 1971.

[15] E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey," *Operations research*, vol. 14, no. 4, pp. 699–719, 1966.

[16] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, "Mot16: A benchmark for multi-object tracking," *arXiv preprint:1603.00831*, 2016.

4.7 Other Publications

4.7.1 MFR 2021: Masked Face Recognition Competition

- **Authors:** Fadi Boutros and Naser Damer and Jan Niklas Kolf and K. Bommanna Raja and Florian Kirchbuchner and Raghavendra Ramachandra and Arjan Kuijper and Pengcheng Fang and Chao Zhang and Fei Wang and David Montero and Naiara Aginako and Basilio Sierra and Marcos Nieto and Mustafa Ekrem Erakin and Uğur Demir and Hazim Kemal and Ekenel and Asaki Kataoka and Kohei Ichikawa and Shizuma Kubo and J Zhang and Mingjie He and Dan Han and S. Shan and Klemen Grm and Vitomir vStruc and Sachith Seneviratne and Nuran Kasthuriarachchi and Sanka Rasnayaka and Pedro C. Neto and Ana F. Sequeira and João Ribeiro Pinto and Mohsen Saffari and Jaime S. Cardoso

- **Booktitle:** Proceedings of the 2021 IEEE International Joint Conference on Biometrics (IJCB)

- **Year:** 2021

- **Publisher:** IEEE

- **Abstract:** This paper presents a summary of the Masked Face Recognition Competitions (MFR) held within the 2021 International Joint Conference on Biometrics (IJCB 2021). The competition attracted a total of 10 participating teams with valid submissions. The affiliations of these teams are diverse and associated with academia and industry in nine different countries. These teams successfully submitted 18 valid solutions. The competition is designed to motivate solutions aiming at enhancing the face recognition accuracy of masked faces. Moreover, the competition considered the deployability of the proposed solutions by taking the compactness of the face recognition models into account. A private dataset representing a collaborative, multi-session, real masked, capture scenario is used to evaluate the submitted solutions. In comparison to one of the top-performing academic face recognition solutions, 10 out of the 18 submitted solutions did score higher masked face verification accuracy.

4.7.2 Accurate 3D Object Detection from Point Cloud Data using Bird's Eye View Representations

- **Authors:** Nerea Aranjuelo and Guus Engels and David Montero and Marcos Nieto and Ignacio Arganda-Carreras and Luis Unzueta and Oihana Otaegui

- **Booktitle:** Proceedings of the 13th International Joint Conference on Computational Intelligence

- **Year:** 2021

- **Publisher:** SciTePress

- **Abstract:** In this paper, we show that accurate 3D object detection is possible using deep neural networks and a Bird's Eye View (BEV) representation of the LiDAR point clouds. Many recent approaches propose complex neural network architectures to process directly the point cloud data. The good results obtained by these methods have left behind the research of BEV-based approaches. However, BEV-based detectors can take advantage of the advances in the 2D object detection field and need to handle much less data, which is important in real-time automotive applications. We propose a two-stage object detection deep neural network, which takes BEV representations as input and validate it in the KITTI BEV benchmark, outperforming state-of-the-art methods. In addition, we show how additional information can be added to our model to improve the accuracy of the smallest and most challenging object classes. This information can come from the same point cloud or an additional sensor's data, such as the camera.

Bibliography

[14] *Tensorflow detection model zoo*. Last viewed: 2018-05-14 (cit. on pp. 22, 24).

[24] *AUTOPILOT H2020 project*. Last viewed: 2018-05-24 (cit. on pp. 20, 21).

[AAJ21] Imran Ahmed, Misbah Ahmad, and Gwanggil Jeon. „Social distance monitoring framework using deep learning architecture to control infection transmission of COVID-19 pandemic“. In: *Sustainable Cities and Society* 69 (2021), p. 102777 (cit. on p. 30).

[AAV12] Igor N. Aizenberg, Naum N. Aizenberg, and Joos Vandewalle. „Multi-Valued and Universal Binary Neurons: Theory, Learning and Applications“. In: 2012 (cit. on p. 1).

[Ahm+20] Imran Ahmed, Misbah Ahmad, Joel Rodrigues, Gwanggil Jeon, and Sadia Din. „A deep learning-based social distance monitoring framework for COVID-19“. In: *Sustainable Cities and Society* 65 (Nov. 2020), p. 102571 (cit. on p. 30).

[Ann13] Mr. Vinod Saroha Annu Malik Anju Sharma. „Greedy Algorithm“. In: *International Journal of Scientific and Research Publications*. Vol. 3. Aug. 2013 (cit. on p. 46).

[AR20] Aqeel Anwar and A. Raychowdhury. „Masked Face Recognition for Secure Authentication“. In: *ArXiv* abs/2008.11104 (2020) (cit. on pp. 40, 42).

[Ara+21] Nerea Aranjuelo, Sara García, Estíbaliz Loyo, Luis Unzueta, and Oihana Otaegui. „Key Strategies for Synthetic Data Generation for Training Intelligent Systems based on People Detection from Omnidirectional Cameras (In press)“. In: *Computers & Electrical Engineering* (2021) (cit. on pp. 29, 30).

[Ben+09] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. „Curriculum learning“. In: *International Conference on Machine Learning*. 2009 (cit. on p. 52).

[Ben+15] Yoshua Bengio, Dong-Hyun Lee, Jörg Bornschein, and Zhouhan Lin. „Towards Biologically Plausible Deep Learning“. In: *ArXiv* abs/1502.04156 (2015) (cit. on p. 1).

[Ben07] Yoshua Bengio. „Learning Deep Architectures for AI“. In: *Found. Trends Mach. Learn.* 2 (2007), pp. 1–127 (cit. on p. 1).

[Bha+17] Umang Bhatt, Shouvik Mani, Edgar Xi, and J. Zico Kolter. „Intelligent Pothole Detection and Road Condition Assessment“. In: *ArXiv e-prints* (Oct. 2017). arXiv: `1710.02595` (cit. on p. 21).

[Bou+21] Fadi Boutros, Naser Damer, Jan Niklas Kolf, et al. „MFR 2021: Masked Face Recognition Competition“. In: *2021 IEEE International Joint Conference on Biometrics (IJCB)* (2021), pp. 1–10 (cit. on p. 44).

[Bow04] K. W. Bowyer. „Face recognition technology: security versus privacy“. In: *IEEE Technology and Society Magazine* 23.1 (2004), pp. 9–19 (cit. on p. 15).

[BWL20a] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020 (cit. on p. 29).

[BWL20b] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. „YOLOv4: Optimal Speed and Accuracy of Object Detection“. In: *ArXiv* abs/2004.10934 (2020) (cit. on p. 5).

[Cao+21] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. „OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43 (2021), pp. 172–186 (cit. on p. 6).

[Car+20] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, et al. „End-to-End Object Detection with Transformers“. In: *ArXiv* abs/2005.12872 (2020) (cit. on p. 52).

[Ces+20] Ivan Cesar, Valentin Solina, Renata Kramberger, and Tin Kramberger. „Enhancing the Performance of Image Preprocessing for Classification and Object Detection“. In: 2020 (cit. on pp. 6, 8).

[Cir+10] Dan C. Ciresan, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. „Deep, Big, Simple Neural Nets for Handwritten Digit Recognition“. In: *Neural Computation* 22 (2010), pp. 3207–3220 (cit. on p. 2).

[Cir+13] Dan C. Ciresan, Alessandro Giusti, Luca Maria Gambardella, and Jürgen Schmidhuber. „Mitosis Detection in Breast Cancer Histology Images with Deep Neural Networks“. In: *Medical image computing and computer-assisted intervention : MICCAI ... International Conference on Medical Image Computing and Computer-Assisted Intervention* 16 Pt 2 (2013), pp. 411–8 (cit. on p. 2).

[CM99] D. Comaniciu and P. Meer. „Mean shift analysis and applications“. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. 1999, 1197–1203 vol.2 (cit. on p. 19).

[CMS12] Dan C. Ciresan, Ueli Meier, and Jürgen Schmidhuber. „Multi-column deep neural networks for image classification“. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition* (2012), pp. 3642–3649 (cit. on p. 2).

[CPS06] Kumar Chellapilla, Sidd Puri, and Patrice Y. Simard. „High Performance Convolutional Neural Networks for Document Processing“. In: 2006 (cit. on p. 2).

[Dec86] Rina Dechter. „Learning While Searching in Constraint-Satisfaction-Problems“. In: *AAAI*. 1986 (cit. on p. 1).

[Dem+21] Alexander Demidovskij, Artyom Tugaryov, Andrej Kashchikhin, et al. „OpenVINO Deep Learning Workbench: Towards Analytical Platform for Neural Networks Inference Optimization“. In: *Journal of Physics: Conference Series* 1828 (2021) (cit. on p. 7).

[Den+09] Jia Deng, Wei Dong, Richard Socher, et al. „Imagenet: A large-scale hierarchical image database“. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255 (cit. on p. 4).

[Den+19a] J. Deng, J. Guo, N. Xue, and S. Zafeiriou. „ArcFace: Additive Angular Margin Loss for Deep Face Recognition“. In: *2019 IEEE/CVF Conf. on Computer Vision and Pattern Recognition*. June 2019, pp. 4685–4694 (cit. on pp. 15, 35).

[Den+19b] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. „ArcFace: Additive Angular Margin Loss for Deep Face Recognition“. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019 (cit. on pp. 40, 42).

[Den+20] Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, et al. *MOT20: A benchmark for multi object tracking in crowded scenes*. 2020 (cit. on p. 30).

[Dos+20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, et al. „An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale“. In: *ArXiv* abs/2010.11929 (2020) (cit. on p. 52).

[EPS17] Hadi Keivan Ekbatani, Oriol Pujol, and Santi Segui. „Synthetic Data Generation for Deep Learning in Counting Pedestrians.“ In: *ICPRAM*. 2017, pp. 318–323 (cit. on pp. 5, 7).

[Erk+09] Zekeriya Erkin, Martin Franz, Jorge Guajardo, et al. „Privacy-Preserving Face Recognition“. In: *Privacy Enhancing Technologies*. Ed. by Ian Goldberg and Mikhail J. Atallah. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 235–253 (cit. on p. 15).

[Est+96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. „A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise“. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, 1996, pp. 226–231 (cit. on pp. 18, 19).

[Eve+10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. „The Pascal Visual Object Classes (VOC) Challenge“. In: *International Journal of Computer Vision* 88.2 (June 2010), pp. 303–338 (cit. on p. 4).

[Gou+20] Jianping Gou, B. Yu, Stephen J. Maybank, and Dacheng Tao. „Knowledge Distillation: A Survey“. In: *ArXiv* abs/2006.05525 (2020) (cit. on p. 52).

[Gou+21] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. „Knowledge distillation: A survey“. In: *International Journal of Computer Vision* 129.6 (2021), pp. 1789–1819 (cit. on p. 7).

[GS05] Faustino J. Gomez and Jürgen Schmidhuber. „Co-evolving recurrent neurons learn deep memory POMDPs“. In: *GECCO '05*. 2005 (cit. on p. 1).

[Guo+16] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. „MS-Celeb-1M: A Dataset and Benchmark for Large-Scale Face Recognition“. In: *ECCV*. Vol. 9907. Oct. 2016, pp. 87–102 (cit. on p. 17).

[He+16a] K. He, X. Zhang, S. Ren, and J. Sun. „Deep Residual Learning for Image Recognition“. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016, pp. 770–778 (cit. on pp. 35, 40).

[He+16b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. „Deep residual learning for image recognition“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778 (cit. on p. 40).

[Ho+03] J. Ho, Ming-Husang Yang, Jongwoo Lim, Kuang-Chih Lee, and D. Kriegman. „Clustering appearances of objects under varying illumination conditions“. In: *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* Vol. 1. 2003, pp. I–I (cit. on p. 19).

[How+17] A. G. Howard, M. Zhu, B. Chen, et al. „MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications“. In: *ArXiv e-prints* (Apr. 2017). arXiv: `1704.04861 [cs.CV]` (cit. on pp. 23, 24).

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. „Long Short-Term Memory“. In: *Neural Computation* 9 (1997), pp. 1735–1780 (cit. on p. 1).

[Hua+17] J. Huang, V. Rathod, C. Sun, et al. „Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors“. In: July 2017, pp. 3296–3297 (cit. on pp. 22, 24).

[ILM67] Aleksei Grigorevich Ivakhnenko, Valentin Grigorevich Lapa, and R. N. Mcdonough. „Cybernetics and forecasting techniques“. In: 1967 (cit. on p. 1).

[JD88] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. USA: Prentice-Hall, Inc., 1988 (cit. on p. 16).

[Jeo+22] Eunji Jeong, Jangryul Kim, Samnieng Tan, Jaeseong Lee, and Soonhoi Ha. „Deep Learning Inference Parallelization on Heterogeneous Processors With TensorRT“. In: *IEEE Embedded Systems Letters* 14 (2022), pp. 15–18 (cit. on p. 7).

[Joc+21] Glenn R. Jocher, Alex Stoken, Jiří Borovec, et al. „ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations“. In: 2021 (cit. on p. 5).

[KKG07] Fatih Kahraman, Binnur Kurt, and Muhittin Gokmen. „Robust Face Alignment for Illumination and Pose Invariant Face Recognition“. In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. 2007, pp. 1–7 (cit. on pp. 5, 7).

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. „ImageNet classification with deep convolutional neural networks“. In: *Communications of the ACM* 60 (2012), pp. 84–90 (cit. on p. 2).

[LBL19] Xin Long, Zongcheng Ben, and Yan Liu. „A Survey of Related Research on Compression and Acceleration of Deep Neural Networks“. In: *Journal of Physics: Conference Series* 1213 (2019) (cit. on p. 7).

[Li+22] Chuyin Li, Lu Li, Hongliang Jiang, et al. „YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications“. In: *ArXiv* abs/2209.02976 (2022) (cit. on p. 5).

[Lin+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, et al. „Microsoft coco: Common objects in context“. In: *European conference on computer vision*. Springer. 2014, pp. 740–755 (cit. on p. 4).

[Lin+17a] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, et al. „Feature pyramid networks for object detection“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125 (cit. on p. 5).

[Lin+17b] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. „Focal loss for dense object detection“. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988 (cit. on pp. 5, 7).

[Lin+18] W. Lin, J. Chen, C. D. Castillo, and R. Chellappa. „Deep Density Clustering of Unconstrained Faces“. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8128–8137 (cit. on p. 36).

[Liu+16] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang. „DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations“. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1096–1104 (cit. on p. 18).

[Liu+21] Ze Liu, Yutong Lin, Yue Cao, et al. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. 2021 (cit. on p. 4).

[Llo82] S. P. Lloyd. „Least squares quantization in PCM“. In: *IEEE Trans. Inf. Theory* 28 (1982), pp. 129–136 (cit. on pp. 16, 18, 19).

[Lu+20] Jiasen Lu, Vedanuj Goswami, Marcus Rohrbach, Devi Parikh, and Stefan Lee. „12-in-1: Multi-Task Vision and Language Representation Learning“. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 10434–10443 (cit. on p. 7).

[LW66] Eugene L Lawler and David E Wood. „Branch-and-bound methods: A survey“. In: *Operations research* 14.4 (1966), pp. 699–719 (cit. on p. 45).

[Maz+18] B. Maze, J. Adams, J. A. Duncan, et al. „IARPA Janus Benchmark - C: Face Dataset and Protocol“. In: *2018 International Conference on Biometrics (ICB)*. Feb. 2018, pp. 158–165 (cit. on p. 17).

[Mil+16] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. „MOT16: A benchmark for multi-object tracking“. In: *arXiv preprint:1603.00831* (2016) (cit. on p. 46).

[Mon+19] David Montero, Nerea Aranjuelo, Orti Senderos, and Marcos Nieto. „BEV Object Tracking for LIDAR-based Ground Truth Generation“. In: *2019 27th European Signal Processing Conference (EUSIPCO)* (2019), pp. 1–5 (cit. on pp. 10, 13).

[Mon+21] David Montero, Luis Unzueta, Jon Goenetxea, et al. „Multi-Stage Dynamic Batching and On-Demand I-Vector Clustering for Cost-effective Video Surveillance“. In: *16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*. 2021 (cit. on pp. 10, 12).

[Mon+22a] David Montero, Naiara Aginako, Basilio Sierra, and Marcos Nieto. „Efficient Large-Scale Face Clustering Using an Online Mixture of Gaussians“. In: *Engineering Applications of Artificial Intelligence* 114 (2022), p. 105079 (cit. on p. 10).

[Mon+22b] David Montero, Marcos Nieto, Peter Leskovský, and Naiara Aginako. „Boosting Masked Face Recognition with Multi-Task ArcFace“. In: *16th International Conference on Signal Image Technology and Internet based Systems (SITIS)*. 2022 (cit. on pp. 10, 12).

[Mon+on] David Montero, Nerea Aranjuelo, Peter Leskovsky, Marcos Nieto, and Naiara Aginako. „Multi-Camera BEV Video-Surveillance System for Efficient Monitoring of Social Distancing“. In: *Multimedia Tools and Applications* (2023 pending publication) (cit. on pp. 10, 11).

[Mon19] David Montero. *face_recognition_TF2*. 2019 (cit. on p. 41).

[MV19] Sparsh Mittal and Shraiysh Vaishay. „A survey of techniques for optimizing deep learning on GPUs“. In: *J. Syst. Archit.* 99 (2019) (cit. on p. 2).

[MWK16] Adam H. Marblestone, Greg Wayne, and Konrad Paul Kording. „Toward an Integration of Deep Learning and Neuroscience“. In: *Frontiers in Computational Neuroscience* 10 (2016) (cit. on p. 1).

[NBK15] S. Nienaber, M Booysen, and R Kroon. „Detecting Potholes Using Simple Image Processing Techniques and Real-World Footage“. In: *34th Annual Southern African Transport Conf. (SATC)*. 2015, pp. 153–164 (cit. on pp. 21, 22).

[Nie+14] Marcos Nieto, Juan Ortega, Oihana Otaegui, and Andoni Cortes. „Optimization of Computer Vision Algorithms in Codesign Methodologies“. In: Sept. 2014 (cit. on p. 8).

[NJW01] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. „On Spectral Clustering: Analysis and an Algorithm“. In: *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*. NIPS'01. Vancouver, British Columbia, Canada: MIT Press, 2001, pp. 849–856 (cit. on p. 19).

[NSM05] E. M. Newton, L. Sweeney, and B. Malin. „Preserving privacy by de-identifying face images“. In: *IEEE Transactions on Knowledge and Data Engineering* 17.2 (2005), pp. 232–243 (cit. on p. 15).

[NSO21] Marcos Nieto, Orti Senderos, and Oihana Otaegui. „Boosting AI applications: Labeling format for complex datasets“. In: *SoftwareX* 13 (2021), p. 100653 (cit. on p. 45).

[OJ04] Kyoungsu Oh and Keechul Jung. „GPU implementation of neural networks“. In: *Pattern Recognit.* 37 (2004), pp. 1311–1314 (cit. on p. 2).

[OWJ18] C. Otto, D. Wang, and A. K. Jain. „Clustering Millions of Faces by Identity“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.2 (Feb. 2018), pp. 289–303 (cit. on pp. 16, 18, 19, 36, 37).

[Par+18] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, et al. „Image Transformer“. In: *International Conference on Machine Learning*. 2018 (cit. on p. 52).

[PSA20] Narinder Singh Punn, Sanjay Kumar Sonbhadra, and Sonali Agarwal. *Monitoring COVID-19 social distancing with person detection and tracking via fine-tuned YOLO v3 and Deepsort techniques*. 2020 (cit. on p. 30).

[PY20] Trong Huy Phan and Kazuma Yamamoto. „Resolving class imbalance in object detection with weighted cross entropy losses". In: *arXiv preprint arXiv:2006.01413* (2020) (cit. on pp. 5, 7).

[RA20] Mahdi Rezaei and Mohsen Azarmi. „DeepSOCIAL: Social Distancing Monitoring and Infection Risk Assessment in COVID-19 Pandemic". In: *Applied Sciences* 10.21 (2020) (cit. on p. 30).

[RMN09] Rajat Raina, Anand Madhavan, and A. Ng. „Large-scale deep unsupervised learning using graphics processors". In: *ICML '09*. 2009 (cit. on p. 2).

[Rum+20] Masuma Akter Rumi, Xiaolong Ma, Yanzhi Wang, and Peng Jiang. „Accelerating sparse CNN inference on GPUs with performance-aware weight pruning". In: *Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques*. 2020, pp. 267–278 (cit. on p. 7).

[Sch15] Jürgen Schmidhuber. „Deep learning in neural networks: An overview". In: *Neural networks : the official journal of the International Neural Network Society* 61 (2015), pp. 85–117 (cit. on p. 1).

[Scu10] D. Sculley. „Web-Scale k-Means Clustering". In: *Proceedings of the 19th International Conference on World Wide Web*. WWW '10. Raleigh, North Carolina, USA: Association for Computing Machinery, 2010, pp. 1177–1178 (cit. on pp. 18, 19).

[Sib73] R. Sibson. „SLINK: An optimally efficient algorithm for the single-link cluster method". In: *The Computer Journal* 16.1 (Jan. 1973), pp. 30–34. eprint: `https://academic.oup.com/comjnl/article-pdf/16/1/30/1196082/160030.pdf` (cit. on pp. 18, 19).

[SK19] Connor Shorten and Taghi M. Khoshgoftaar. „A survey on Image Data Augmentation for Deep Learning". In: *Journal of Big Data* 6 (2019), pp. 1–48 (cit. on pp. 5, 7).

[SKP15] F. Schroff, D. Kalenichenko, and J. Philbin. „FaceNet: A unified embedding for face recognition and clustering". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 815–823 (cit. on p. 15).

[SOJ18] Y. Shi, C. Otto, and A. K. Jain. „Face Clustering: Representation and Pairwise Constraints". In: *IEEE Transactions on Information Forensics and Security* 13.7 (July 2018), pp. 1626–1640 (cit. on p. 36).

[Son+19] Yanan Song, Quan-Ke Pan, Liang Gao, and Biao Zhang. „Improved non-maximum suppression for object detection using harmony search algorithm". In: *Applied Soft Computing* 81 (2019), p. 105478 (cit. on pp. 6, 8).

[SWC20] Atefeh Sohrabizadeh, Jie Wang, and Jason Cong. „End-to-End Optimization of Deep Learning Applications". In: *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. FPGA '20. Seaside, CA, USA: Association for Computing Machinery, 2020, pp. 133–139 (cit. on pp. 6, 7).

[Sze+16] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. „Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: (Feb. 2016). arXiv: `1602.07261` (cit. on p. 24).

[Sze+17] Vivienne Sze, Yu-hsin Chen, Tien-Ju Yang, and Joel S. Emer. „Efficient Processing of Deep Neural Networks: A Tutorial and Survey“. In: *Proceedings of the IEEE* 105 (2017), pp. 2295–2329 (cit. on p. 2).

[Ten17] A. TensorFlow. „Implementation of control flow in TensorFlow“. In: *TensorFlow Whitepaper* (2017) (cit. on p. 34).

[TL19] Mingxing Tan and Quoc V. Le. „EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks“. In: *ArXiv* abs/1905.11946 (2019) (cit. on p. 5).

[TPL20] Mingxing Tan, Ruoming Pang, and Quoc V Le. „Efficientdet: Scalable and efficient object detection“. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 10781–10790 (cit. on p. 4).

[Vas+17] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, et al. „Attention is All you Need“. In: *ArXiv* abs/1706.03762 (2017) (cit. on p. 52).

[Wan+17] X. Wang, Ross B. Girshick, Abhinav Kumar Gupta, and Kaiming He. „Non-local Neural Networks“. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2017), pp. 7794–7803 (cit. on p. 52).

[Wan+18] H. Wang, Y. Wang, Z. Zhou, et al. „CosFace: Large Margin Cosine Loss for Deep Face Recognition“. In: *2018 IEEE/CVF Conf. on Computer Vision and Pattern Recognition*. June 2018, pp. 5265–5274 (cit. on p. 34).

[Wan+19] Zhongdao Wang, Liang Zheng, Yali Li, and Shengjin Wang. „Linkage Based Face Clustering via Graph Convolution Network“. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 1117–1125 (cit. on pp. 16, 18–20, 36, 37).

[WBL22] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. 2022 (cit. on p. 4).

[Whi+17] C. Whitelam, E. Taborsky, A. Blanton, et al. „IARPA Janus Benchmark-B Face Dataset“. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. July 2017, pp. 592–600 (cit. on p. 17).

[WSH20] Xiongwei Wu, Doyen Sahoo, and Steven CH Hoi. „Recent advances in deep learning for object detection“. In: *Neurocomputing* 396 (2020), pp. 39–64 (cit. on p. 4).

[Xia+20] Youzi Xiao, Zhiqiang Tian, Jiachen Yu, et al. „A review of object detection based on deep learning“. In: *Multimedia Tools and Applications* 79.33 (2020), pp. 23729–23791 (cit. on p. 4).

[Xie+20] Zihao Xie, Li Zhu, Lin Zhao, et al. „Localization-aware channel pruning for object detection“. In: *Neurocomputing* 403 (2020), pp. 400–408 (cit. on pp. 6, 7).

[XZ17] Jia Xiang and Gengming Zhu. „Joint face detection and facial expression recognition with MTCNN“. In: *2017 4th international conference on information science and control engineering (ICISCE)*. IEEE. 2017, pp. 424–427 (cit. on p. 5).

[Yan+19] L. Yang, X. Zhan, D. Chen, et al. „Learning to Cluster Faces on an Affinity Graph“. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 2293–2301 (cit. on pp. 18, 19).

[Yan+20a] Dongfang Yang, Ekim Yurtsever, Vishnu Renganathan, Keith A. Redmill, and Ümit Özgüner. *A Vision-based Social Distancing and Critical Density Detection System for COVID-19*. 2020 (cit. on p. 30).

[Yan+20b] L. Yang, D. Chen, X. Zhan, et al. „Learning to Cluster Faces via Confidence and Connectivity Estimation“. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 13366–13375 (cit. on pp. 18, 19).

[Yiz95] Yizong Cheng. „Mean shift, mode seeking, and clustering“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17.8 (1995), pp. 790–799 (cit. on p. 19).

[YMA21] J. Javier Yebes, David Montero, and Ignacio Arriola. „Learning to Automatically Catch Potholes in Worldwide Road Scene Images“. In: *IEEE Intelligent Transportation Systems Magazine* 13 (2021), pp. 192–205 (cit. on pp. 10, 11).

[Zha+16a] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. „Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks“. In: *IEEE Signal Processing Letters* 23.10 (Oct. 2016), pp. 1499–1503 (cit. on p. 39).

[Zha+16b] Z. Zhang, P. Luo, C. Loy, and X. Tang. „Learning Deep Representation for Face Alignment with Auxiliary Attributes“. In: *IEEE TPAMI* 38.5 (May 2016), pp. 918–930 (cit. on p. 35).

[Zha+18] Xiaohang Zhan, Ziwei Liu, Junjie Yan, Dahua Lin, and Chen Change Loy. „Consensus-Driven Propagation in Massive Unlabeled Data for Face Recognition“. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018 (cit. on pp. 18, 19).

[Zha+19a] Zhi Zhang, Tong He, Hang Zhang, et al. *Bag of Freebies for Training Object Detection Neural Networks*. 2019 (cit. on p. 5).

[Zha+19b] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. „Object detection with deep learning: A review“. In: *IEEE transactions on neural networks and learning systems* 30.11 (2019), pp. 3212–3232 (cit. on p. 4).

[Zha+88] Wei Zhang, Jun Tanida, Kazuyoshi Itoh, and Yoshiki Ichioka. „Shift-invariant pattern recognition neural network and its optical architecture“. In: *Proceedings of annual conference of the Japan Society of Applied Physics*. 1988, pp. 2147–2151 (cit. on p. 1).

[Zha+90] W Zhang, Kazuyoshi Itoh, Jun Tanida, and Yoshiki Ichioka. „Parallel distributed processing model with local space-invariant interconnections and its optical architecture.“ In: *Applied optics* 29 32 (1990), pp. 4790–7 (cit. on p. 1).

[Zop+20] Barret Zoph, Ekin D Cubuk, Golnaz Ghiasi, et al. „Learning data augmentation strategies for object detection“. In: *European Conference on Computer Vision*. Springer. 2020, pp. 566–583 (cit. on p. 30).

[ZZL19] Pengyi Zhang, Yunxin Zhong, and Xiaoqiong Li. „SlimYOLOv3: Narrower, Faster and Better for Real-Time UAV Applications“. In: *CoRR* abs/1907.11093 (2019) (cit. on p. 30).

List of Figures

List of Tables