Scuola universitaria professionale della Svizzera italiana
Dipartimento tecnologie innovative

SUPSI **ISIN TECHNICAL REPORT**

Decentralized Smart City Surveillance

Data Processing Middleware for On-Premises Video Data Analysis

Author(s)

Amos Brocco[1]
Vanni Galli[2]

Technical Report Number

2018-1

Date

January 22, 2018

[1]amos.brocco@supsi.ch
[2]vanni.galli@supsi.ch

Abstract

Surveillance cameras are nowadays pervasive elements in our cities: they act as a deterrent against illegal activities or traffic violations, they help investigating crimes, and they can provide useful traffic and people analytics. Collecting video streams from multiple sources, possibly in real-time, poses a number of ethical, legal and technical challenges. As engineers we mainly focus on the latter issue, leaving the first two to politicians and lawyers. On the one hand, we recognize that the infrastructure needed to stream data needs to be sufficiently capable, robust and scalable; on the other hand, we know that plentiful of raw processing power and a vast storage capacity are required to analyze multiple video streams in real time and save the resulting information. In this paper we aim attention at those technical issues and present an on-going project focused at providing real-time information by means of a flexible processing infrastructure. We tackle the robustness concern through a decentralized middleware solution, and we aim at improving the scalability of the system by using machine learning techniques to extrapolate structured knowledge as soon as possible, so that only the relevant data needs to be transmitted further in the network and stored on remote servers. This approach enables surveillance operators to quickly inquire the system by means of morphological or color search criteria instead of watching hundreds of video streams, then select and automatically track interesting items within different video streams.

1 Introduction

The concept of a *smart city* underlies not only a strategy of implementing innovative concepts to public services but also integrating novel ideas to improve quality of life [1]. In this regard, an important aspect is data analytics : gathering and extracting information from sensors provides a better understanding of the dynamics of a city, and supports decisions for improving energy efficiency, reduce traffic or increase security [2]. Unfortunately public monitoring also raises several concerns [3], as sensors and cameras gather large amounts of data all day long, uninterrupted for months and years. Data might also be complex or time-consuming to analyze. This concern is exacerbated even more if one considers video surveillance data: from a technical perspective, video streams easily consume bandwidth, computational power and storage space; from a practical standpoint, accurate analysis of hundreds of video streams cannot be efficiently performed by a human operator.
With the advancement of technology and the astounding computational power of modern computers, automated or intelligent video analysis [4] have become very active research areas [5], solving problems ranging from people or vehicle detection and tracking [6, 7, 8], to face [9, 10] or license plate recognition [11]. Analysis at higher levels is also possible, for example to recognize people's activity [12] or detect abnormal situations [13]. Video analysis is usually a multi-step process: algorithms for detecting and tracking objects have first to deal with complex situations, such as overlapping and occlusions [14], subsequently artificial intelligence and machine learning techniques can be employed for recognizing or classifying acquired data [15]. Depending on the level of the analysis, those tasks can be very demanding, and require GPU-Accelerated solutions to achieve real-time performance [16, 17].
Beside algorithms, video surveillance systems also need a back-end infrastructure supporting both the acquisition phase (streaming video from a multitude of cameras) and the storage of processed data, should implement notification mechanisms to alert users about specific events, and must provide front-ends for accessing and searching stored information. In order to afford the required computational power and storage space, there have been proposals to move video processing into the cloud, namely through VSaaS (Video Surveillance as a Service) platforms [18, 19]. These solutions relieve operators from the burden of managing complex setups and storing large amounts of sensitive information, but are not always applicable: infrastructural or interoperability issues with the

existing surveillance system could constitute roadblocks on the migration path between a traditional platform and the cloud. Compared to on-premises solutions, cloud solutions might also raise legal concerns about privacy and security [20], and are therefore not viable solution if video streams are not allowed to be sent to a third-party.
In an effort to prove that automated video surveillance can still be efficiently deployed without necessarily relying on cloud resources, this paper presents an on-going project that aims at implementing a decentralized automated video surveillance system. By employing image analysis and machine learning techniques, relevant data is extrapolated as structured information at or close to each acquisition device; compared to a centralized video analysis system, the proposed solution avoids a single-point of failure and allows for very little data to be sent to remote storage servers. We argue that such a solution will provide flexibility, robustness, and scalability. Compared to traditional manual video analysis, data structuring enables surveillance operators to quickly inquire the system by means of morphological or color search criteria (instead of watching hundreds of video streams), then select and automatically track interesting items within different video streams.

2 Related work

Automated video surveillance systems is related to several different problems: research on this topic typically focuses on specific challenges such as object detection, multi-object tracking, pedestrian detection or movement analysis. In this section we first provide an insight on the main problems and the proposed solutions from an algorithmic and technical point of view; subsequently we will focus on surveillance platforms available on the market.
One of the fundamental requirements for achieving automated video surveillance is object segmentation, employed for separating objects from the background [21]. The system must be typically capable of resolving multiple objects within the same scene, either pedestrians or vehicles. Separation can be achieved using background substraction techniques [22] or motion information [23].
Subsequently, detected objects can be analyzed to extract features such as color, size, orientation or speed. Objects can also can classified using machine learning [24], for example to distinguish between human and vehicles or to determine vehicle class (car, truck, motorcycle, etc.). Each step provides additional information which can be useful for subsequent analysis and for solving the problem of object re-identification, that is the capability of recognizing the same object across multiple cameras [25]. Higher level algorithms can be also implemented to figure out people's behaviour [26] or traffic flows [27].
Computer vision algorithms are just one of the pieces needed for of creating an automated video surveillance system. A fundamental role is played by the infrastructure needed for transmitting and storing data. In this regard a lot of research has taken place in the field of Visual Sensor Networks (VSN) [28], which are distributed architectures, comprised of a large number of low-power camera nodes, for capturing, processing and extracting relevant visual information through autonomous collaboration. Middlewares for VSNs are typically proposed for autonomous visual surveillance or vehicle traffic monitoring [29, 30].
Another critical issue are bandwidth requirements, which might easily hinder the scalability of a surveillance system. In this sense, the problem brings into consideration not only the available network bandwidth but also the desired video quality and the computational complexity of the encoding and decoding algorithms[31]. In the context of video surveillance adaptive compression techniques [32] as well as ad hoc algorithms [33] have been proposed.

On the market, comprehensive platforms such as IBM[3] or Bosch[4] intelligent video analytics, focus on providing fully integrated solutions tailored to customer's needs to detect and analyze video streams. More recently, some vendors, such as Agent Vi [5], introduced cloud based solutions to provide Video Surveillance as a Service (VSaaS) [18].
Even through this brief review of related works we can see how automated video surveillance is a relevant topic both from a research perspective and from a commercial point of view. In this regard, our proposition is to build upon the concept of VSN, and base our solution on a decentralized middleware architecture which exploits the increased computing power and lower cost of current embedded devices.

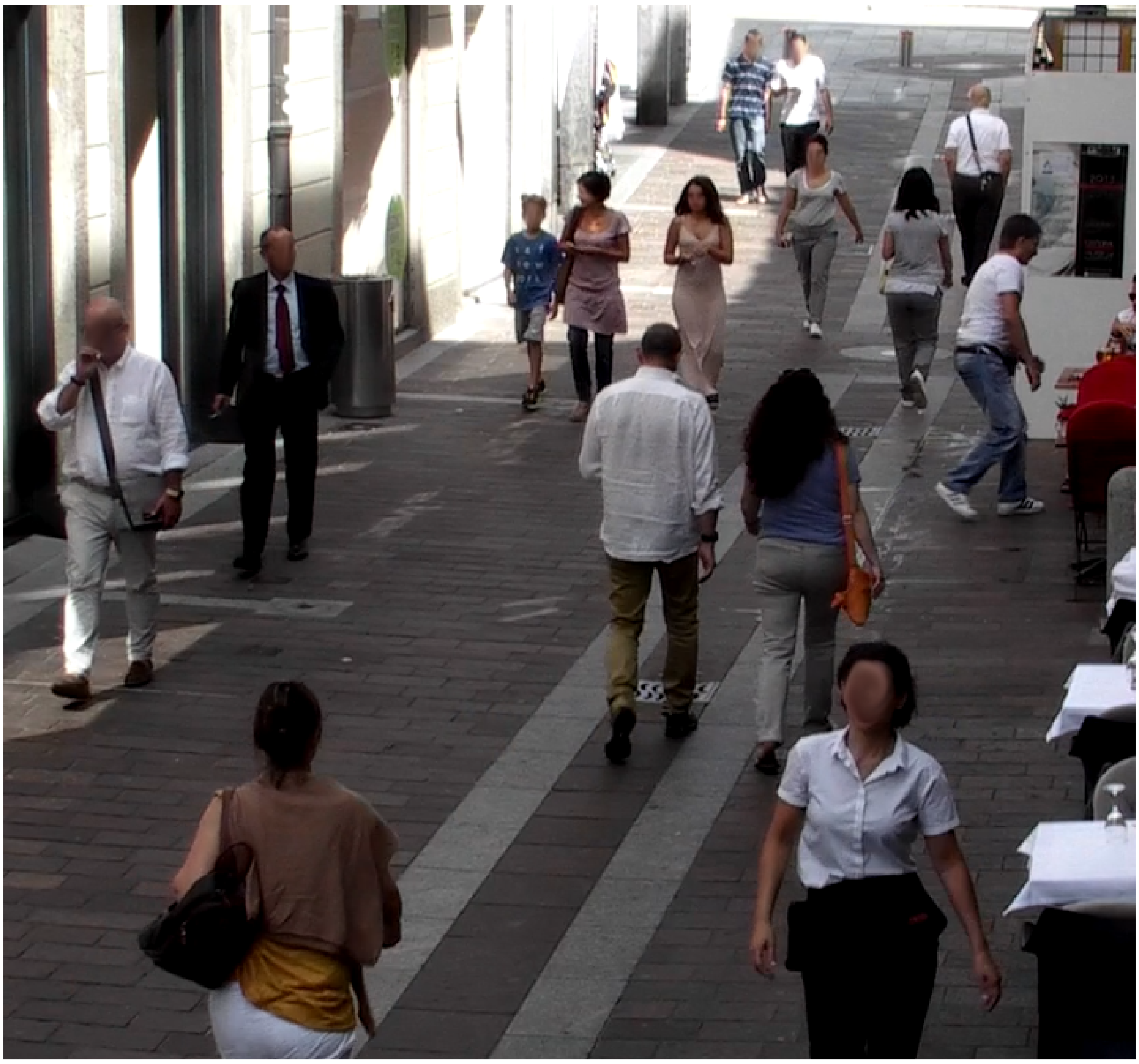


Figure 1: Example frame captured from a surveillance camera (faces have been blurred on purpose)

[3]http://www-03.ibm.com/software/products/it/intelligent-video-analytics
[4]http://ipp.boschsecurity.com/en/tools/video-tools/video-analytics-overview/intelligent-video-analysis-1
[5]https://www.agentvi.com/

3 Research and development context

The middleware platform presented in this paper is being developed in the context of an international project in collaboration with two industrial partners (one headquartered in Switzerland, the other in Lithuania). Research and development are financially supported by the Swiss Commission for Technology and Innovation CTI [6] and the European Eurostars program [7]. The project aims at delivering an hardware and software platform for processing and managing data gathered by surveillance cameras. The first end-user and testing partner in this project is a municipality with a population of 70000 located in Switzerland.
The city is monitored using about 250 IP connected cameras, pointed at different streets and places (Figure 1). Currently the surveillance platform is provided by Avigilon™ [8]: video streams are available in real-time over the network, whereas stored data can be accessed through the platform SDK. In the event of a crime or other illegal activity, local police has access to the last 100 hours of recorded video from each camera: since this process is long and tedious, the envisioned solution needs to introduce computer-assisted intelligent solutions to scout for relevant information by analyzing people and vehicles density flow patterns in public areas, index and manage objects of interest, and notify the user if abnormal behaviors are detected. Legally wise the system is not allowed to store data for a long time, therefore stored information must be periodically deleted: this is a mandatory requirement that needs to be considered for the new system too. Concerning the network infrastructure provided by the municipality, the main requirement imposed to this project is limiting bandwidth consumption, because a significant increase of the generated traffic can easily choke the network or prevent further expansion of the surveillance system.

4 Decentralized middleware solution

Infrastructure wise, the requirements of the project are low bandwidth consumption, robustness and scalability: the system must limit the amount of data exchanged on the network, should be resilient to local failures (either network or processing issues) and must be able to accommodate additional surveillance cameras. A single central server or a cloud-based service for processing all video streams would likely be the easiest solution, but such a system would need to be both well connected (in terms of bandwidth) and sufficiently powerful (if one considers an on-premises solution). As an example, in the current deployment scenario cameras provide either H.264 or MJPEG video streams with a resolution up to 1920 by 1080 pixels at 30 fps: as such, the bit rate required for realtime transmission from each camera ranges from 2 to 20 Mbps. A centralized solution would act as bottleneck and would become a single point of failure: moreover deploying additional cameras would likely introduce considerable costs due to the increased processing power and bandwidth requirements. To limit these issues, our approach involves the use of a distributed data processing infrastructure: we propose a decentralized middleware, where network-intensive video data transmission is limited and where processing is performed as close as possible to the acquisition devices (i.e. cameras). The advantages of such a solution are twofold: on the one side, bandwidth requirements are reduced because only pre-processed relevant information is sent toward storage servers; on the other side, the system is inherently more robust and scalable, because failures (either in the video acquisition, data transmission or processing phases) only affect specific surveillance areas.

[6] https://www.kti.admin.ch
[7] https://www.eurostars-eureka.eu/
[8] http://avigilon.com

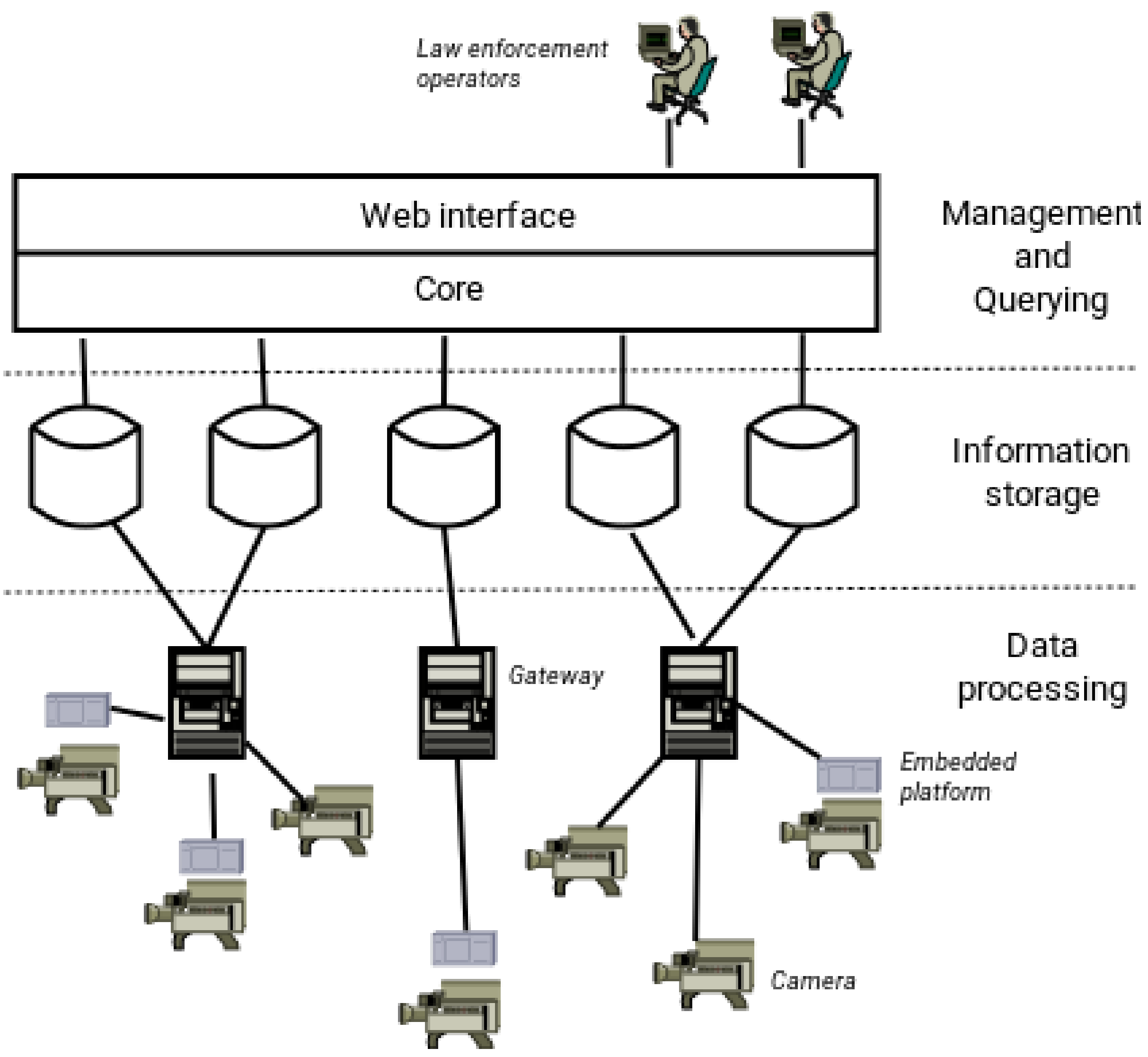


Figure 2: Overview of the middleware

4.1 Overview

The middleware is divided into three logical layers (Figure 2): the data processing layer, the information storage layer, and the query and management layer. The first and the second layers starting from the bottom (where surveillance cameras are connected to), operate in a decentralized manner: decentralization allows each component to act as an independent entity, increasing robustness and scalability. For management and user access purposes, the uppermost layer provides a web interface and implements algorithms for querying the system and for setting alarms and notifications.

4.2 Data processing

Video data stream processing is achieved by means of modular pipelines whose configuration can be tailored to the computational power, storage and connectivity of the target machine. In current implementation, pipelines are implemented on a custom framework written in C++ running on

GNU/Linux. For video processing the system depends on libVLC [9] and the OpenCV [10] libraries; the pipeline framework makes use of multi-threading and seamlessly supports multi-core hardware and graphic accelerators.
Pipelines enable a flexible configuration of the system: if necessary, raw computational power on a node can be traded for additional bandwidth by splitting the pipeline across multiple nodes to distribute the load or simply to perform computationally expensive tasks on a more suitable machine. Since one of the goals of this project is reducing the cost for network infrastructure, pipelines are constructed in such a way as to extract relevant information (and metadata) at an early stage in the processing workflow and close to the acquisition device. Accordingly, bandwidth requirements drop sharply compared to a centralized solution: for example, instead of continuously streaming videos from a camera pointed on a footpath to a storage and processing server, the system just needs to send single frames with detected pedestrians, each one tagged with its direction of movement and speed, summing up to an average size of just few hundreds kbytes.
Because each node can be configured independently, the data processing infrastructure can be composed of a heterogeneous set of hardware platforms: close to each camera, low-cost embedded computers (such as Raspberry Pi[11]) can be used to perform object tracking using simple algorithms, or can provide speed or direction estimations; farther away from the cameras, more powerful machine called gateway nodes (using Intel©i7™ processors coupled with NVIDIA™ GPUs) are tasked with performing computationally expensive tasks such as classification on the data coming from embedded systems. Communication between modules in the same pipeline is based on ZeroMQ [12]: metadata attributes are encoded in a JSON document and sent along with the frame containing the object and a background mask. To simplify the creation and the deployment of pipelines several support tools have been developed: an operator can compose a processing pipeline and assign nodes to different machines using a visual editor (Figure 3).

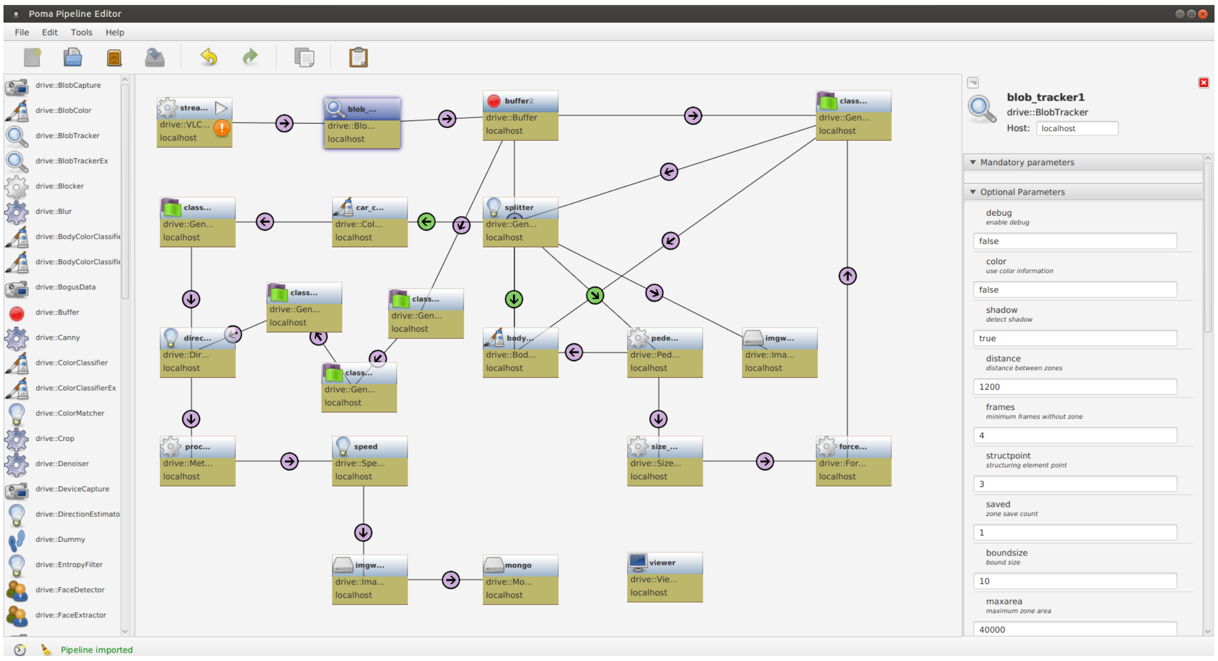
4.2.1 Object tracking

The first step in each video processing pipeline is detection and tracking of moving objects, which need to be isolated from the background. In the current implementation we only consider static surveillance cameras, for which different trackers have been implemented: a simple tracker based on background subtraction, one based on motion templates [34], one employing a meanshift algorithm [35], and another one based on kernelized correlation filters (KCF)[36]. The choice between these algorithms depends on the usage scenario (people or vehicle tracking, camera position, etc.). Tracking modules process camera frames and send results (with metadata such as direction and speed) to a classification module.

4.2.2 Object classification

Classification is initially performed to distinguish between humans (single person or group) and vehicles (cars, motorcycles, vans and trucks); depending on the classification result the object is forwarded to a different path, in order to extract features specific to each class (for example, the sex of a person). The classification module employs machine learning and depends on the Caffe [13]

[9]https://www.videolan.org/
[10]http://opencv.org/
[11]https://www.raspberrypi.org/
[12]http://zeromq.org/
[13]http://caffe.berkeleyvision.org/



blob_tracker1
drive::BlobTracker
Host: localhost

Mandatory parameters

Optional Parameters

debug
enable debug
false

color
use color information
false

shadow
detect shadow
true

distance
distance between zones
1200

frames
minimum frames without zone
4

structpoint
structuring element point
3

saved
zone save count
1

boundsize
bound size
10

maxarea
maximum zone area
40000

Figure 3: Visual editor window with a typical processing pipeline

deep learning framework. Being a computationally expensive task, classification is not suited for low power of embedded systems, but is only executed on gateway nodes with a GPU accelerator.
Multiple classification modules can be added to a pipeline to refine the information stored along with each object. As an example, vehicle classification can be followed by make and model classification, whereas human classification can entail sex or age classifications. Each classifier requires precisely trained models, and produces an output vector which contains the probabilities of the object of belonging to a specific class.

4.2.3 Color matching

Classified items are further processed to determine the dominant color. Colors are a distinct characteristic of an object and provide a useful search criteria to filter out irrelevant information. In the case of detected vehicles, the color matching process considers the whole image, whereas for humans the matching operation produces separate results for the upper and lower halves of the picture.
Precise color matching depends on several factors, first and foremost the lighting conditions of the captured scene. Moreover, determining the dominant color of an image is heavily influenced by the capabilities of a camera to faithfully encode the acquired image, the dynamic range of the camera sensor, sensor noise, white balance, compression artifacts, etc. The same scene captured by different cameras can easily present large differences in color reproduction.
To tackle this problem we employ a color matching algorithm based on template palettes: for each considered color class a template image with different shades of the same color is produced. Templates can vary between each camera and help minimize matching errors caused by color variations due lighting conditions. Captured images and templates are processed to produce histograms representing the distribution of pixel colors in the image: by comparing these histograms it is possible to produce a distance value for each color class. Distance values are grouped into a vector, where the largest components determine the dominant colors in the image. This vector is added to metadata

information before sending the item forward in the pipeline.

4.3 Information storage

At the end of a processing pipeline, data items are sent to a database for indexing and further analysis. In order to create a flexible data structure which can easily accommodate future changes, storage is based on a the non-relational database MongoDB[14]. For each item, the database stores a JSON style document with all the metadata, as well as the captured image and the background mask. Because there is no direct connection between the upper and the data processing layers, the database also manages configuration data for each camera (name, location, frame rate, etc.) and for each processing node. Depending on the size of the network, one or multiple instances of MongoDB can be deployed in the system in order to achieve horizontal scaling through sharding. To cope with network or database failures, each node can keep data on its local disk while waiting for the connection with the storage server to be restored.

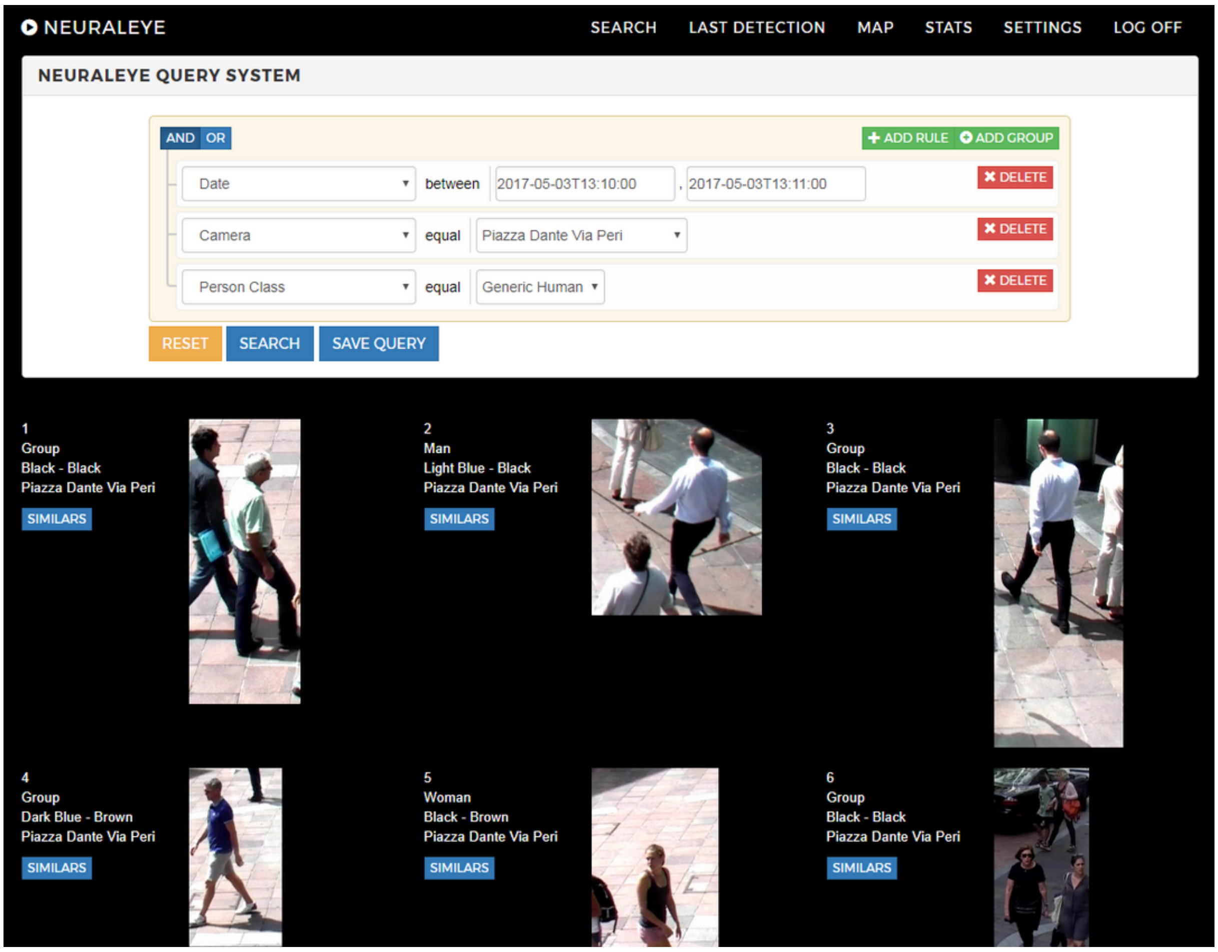
4.4 Core system and querying interface

From an operational point of view, the system needs to assist law enforcement agencies in the task of video surveillance, by making the system easy to configure and maintain, and by making relevant information easily accessible. Beside decentralized processing and storage components, the proposed middleware thus comprises a core system which is tasked with monitoring the infrastructure for faults and failures, deploying camera and remote node configurations and maintaining databases. The core system is also responsible for exposing a web interface which allows end-users to browse stored data, perform search queries and setup alarms (Figure 4). Multiple filter clauses can be combined within the same search query using boolean operators: each clause will filter results according to peculiar characteristics of the searched item, such as geo-localization data (based on camera position), date and time, color, or classification results.
The core system is implemented in C# (ASP.NET)/C++ and can operate independently from the processing platform. Communication between data processing nodes and the core system is strictly asynchronous and is mediated by the storage layer: node configurations and status reports are kept on a MongoDB database, and it is up to each camera node to periodically check for updates. Since MongoDB has no native automated task management, in order to fulfill legal requirements, the core system is also responsible for deleting all the data collected more than 100 hours before.

4.4.1 Object re-identification

An important aspect of surveillance is the ability to track subjects (people or vehicles) across different cameras (also called people re-identification) in order to infer direction of movement and therefore their (probable) future location. Object re-identification is triggered by a human operator through the querying interface, and is performed within the core system. The task involves comparing a selected item with (possibly) all other images in the database (which would introduce a complexity of at least $O(N^2)$).
Concerning vehicles, re-identification can be generally achieved through license plate recognition. Unfortunately this is not always possible, for example if cameras provide a low resolution video stream. Furthermore, such a technique will fail if the license plate is hidden behind other objects. To reliably track any object, including pedestrians or people on a bicycle, across multiple cameras other techniques are therefore needed. In our project we tackle this problem by implementing a multimodal

[14] https://www.mongodb.com



Date between 2017-05-03T13:10:00 , 2017-05-03T13:11:00

Camera equal Piazza Dante Via Peri

Person Class equal Generic Human

1
Group
Black - Black
Piazza Dante Via Peri

2
Man
Light Blue - Black
Piazza Dante Via Peri

3
Group
Black - Black
Piazza Dante Via Peri

4
Group
Dark Blue - Brown
Piazza Dante Via Peri

5
Woman
Black - Brown
Piazza Dante Via Peri

6
Group
Black - Black
Piazza Dante Via Peri

Figure 4: Querying interface

fusion algorithm to combine different measurements and obtain a weighted similarity score. To reduce the complexity of this task, a pruning step is also employed: data filtering algorithms based on geo-localization, object classes and other specific heuristics (for example, projected distance traveled based on speed and direction) are used to create a reduced comparison set upon which a similarity analysis can be performed in a reasonable amount of time.
Once the comparison set is defined, a feature comparison algorithm can be used. In this project we employ the SURF algorithm [37] to extract and compare basic features from each image. This algorithm works on a grayscale copy of the image and detects matching features between two images. Using a distance calculation a numerical value representing how different the images are, is computed: the lower this value, the more similar the two images are (a value of zero is expected when comparing an image with itself).
Unfortunately the similarity between two object computed by the SURF algorithm heavily depends on the camera angle: for example, two images of the same person, one taken from the front, one from the back, will likely result in a large SURF distance and deceive the algorithm. Therefore this value alone is not enough for the purpose of re-identification. To improve on this, we make use of other comparable values obtained from the processing phase, namely the resulting vectors of the classification and color matching steps. Vectors are compared using the euclidean distance, and the weighted sum of all distances is used to determine the similarity between two images (hence the term *multimodal*). Since the re-identification algorithm is probabilistic, results might still contain errors. More specifically, given a target image object by the user, the system can either report completely unrelated objects as similar (false positive) or ignore a different picture of the target object (missed detection). This problem cannot be fully eradicated, but in order to deploy a robust re-identification algorithm both false positives and missed detections need to be minimal.
In order to validate our approach we employed the 3DPes dataset [38]. As shown in Figure 5, the SURF algorithm alone is not enough to obtain satisfactory results when it comes to object re-identification. Accordingly, the DET curve shows that it is very difficult to obtain a low missed detection rate along with a false positive rate. However, by combining the similarity score of the SURF algorithm with those of the color matching module and the classificator, we were able to significantly improve the re-identification rate with just 25% in both the missed detection rate and the false positive rate. These results, combined with an early filtering of the data, enable for an reasonable retrieval of different images of the same individual even when captured from different cameras and angles.

5 Conclusions and future work

In this paper we presented an on-going project aimed at building an intelligent surveillance solution for smart cities. The proposed approach is based on a middleware which delegates the video processing tasks to decentralized nodes installed in proximity of each camera in order to limit bandwidth requirements. Video analysis is achieved through a modular data processing pipeline which can be splitted across multiple nodes, depending on the computational power of each machine. The processing phase produces structured information which is subsequently stored on document-based databases and can be queried by means of morphological or color search criteria. The system makes use of several neural-network classifiers to organize data into categories for faster retrieval. Furthermore, an object re-identification mechanism based on a multimodal fusion algorithm has been implemented to assist human operators in matching subjects (people or vehicles) across different cameras views: results show that satisfactory results can only be obtained by reducing the size of the comparison set and by merging results from different comparison measures.

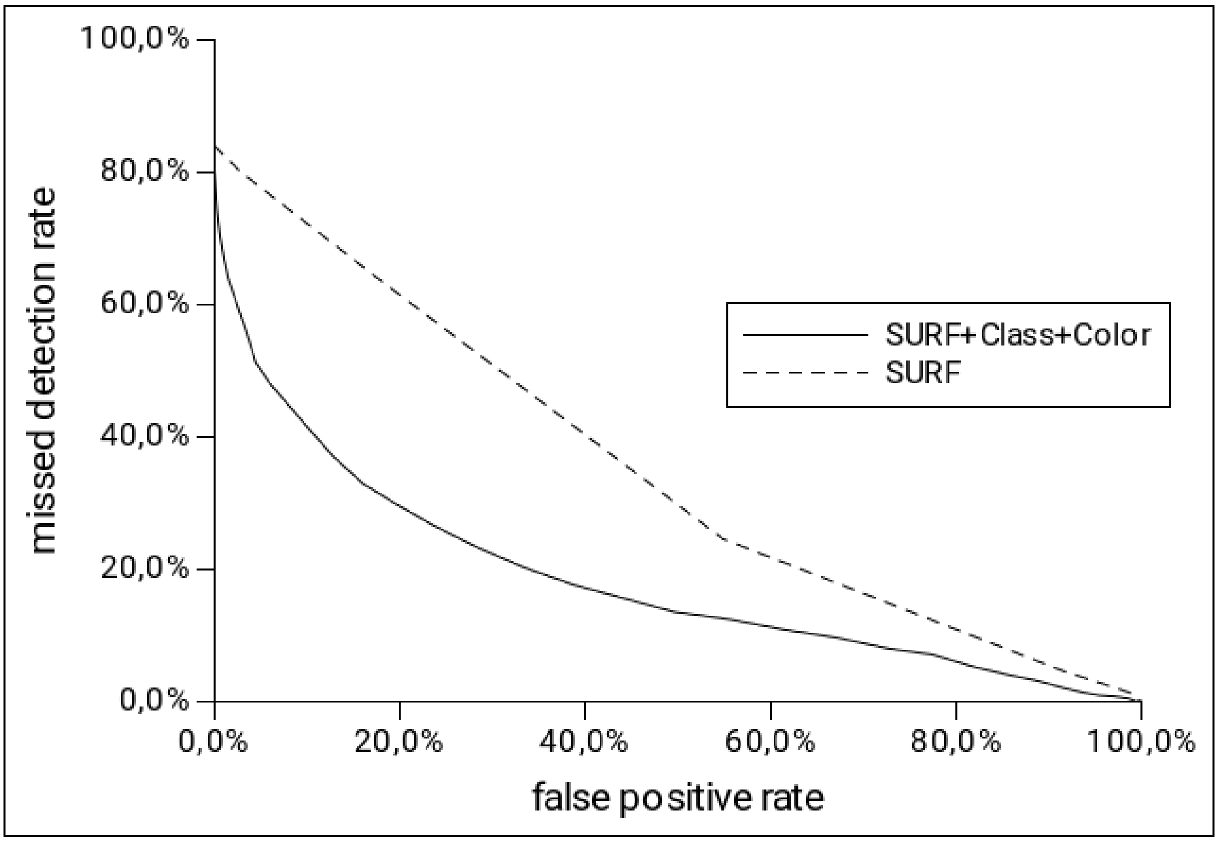


Figure 5: Detection error tradeoff (DET) curve for the object re-identification step

This project has not yet reached its conclusion: even though several interesting results have already been obtained, there are still some issues that need be investigated. The scalability and robustness of the solution needs to be thoroughly verified with field tests. In order to improve object classification, a feedback mechanism and an automatic retraining of each neural network model would need to be implemented. The idea is to initially provide neural-networks with a generic model which will be updated when the system is deployed or when a new camera is added. Another future work is the integration of mobile cameras, such as vehicle mounted ones: this step introduces further challenges in the data processing part of the middleware, specifically for the object detection and tracking phases. Finally, the system currently lacks an automatic behavior recognition mechanism, which needs to be implemented to trigger notifications when abnormal situations are detected.

References

[1] Paolo Neirotti, Alberto De Marco, Anna Corinna Cagliano, Giulio Mangano, and Francesco Scorrano. Current trends in smart city initiatives: Some stylised facts. *Cities*, 38:25 – 36, 2014.

[2] Eiman Al Nuaimi, Hind Al Neyadi, Nader Mohamed, and Jameela Al-Jaroodi. Applications of big data to smart cities. *Journal of Internet Services and Applications*, 6(1):25, Dec 2015.

[3] Liesbet van Zoonen. Privacy concerns in smart cities. *Government Information Quarterly*, 33(3):472 – 480, 2016. Open and Smart Governments: Strategies, Tools, and Experiences.

[4] H. Liu, S. Chen, and N. Kubota. Intelligent video systems and analytics: A survey. *IEEE Transactions on Industrial Informatics*, 9(3):1222–1233, Aug 2013.

[5] P.K. Atrey, M.S. Kankanhalli, and A. Cavallaro, editors. *Intelligent Multimedia Surveillance*. Springer, 2013.

[6] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4), December 2006.

[7] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):743–761, April 2012.

[8] P. K. Bhaskar and S. P. Yong. Image processing based vehicle detection and tracking method. In *2014 International Conference on Computer and Information Sciences (ICCOINS)*, pages 1–5, June 2014.

[9] Z. Lei, C. Wang, Q. Wang, and Y. Huang. Real-time face detection and recognition for video surveillance applications. In *2009 WRI World Congress on Computer Science and Information Engineering*, volume 5, pages 168–172, March 2009.

[10] Joshila Grace. L. K and K. Reshmi. Face recognition in surveillance system. In *2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, pages 1–5, March 2015.

[11] S. Du, M. Ibrahim, M. Shehata, and W. Badawy. Automatic license plate recognition (alpr): A state-of-the-art review. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(2):311–325, Feb 2013.

[12] T. Subetha and S. Chitrakala. A survey on human activity recognition from videos. In *2016 International Conference on Information Communication and Embedded Systems (ICICES)*, pages 1–7, Feb 2016.

[13] A. Tokta and A. K. Hocaoğlu. Abnormal crowd behavior detection in video systems. In *2016 24th Signal Processing and Communication Application Conference (SIU)*, pages 697–700, May 2016.

[14] J. Pan and B. Hu. Robust occlusion handling in object tracking. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007.

[15] Xin Zhang, Yee-Hong Yang, Zhiguang Han, Hui Wang, and Chao Gao. Object class detection: A survey. *ACM Comput. Surv.*, 46(1):10:1–10:53, July 2013.

[16] J. M. Ready and C. N. Taylor. Gpu acceleration of real-time feature based algorithms. In *Motion and Video Computing, 2007. WMVC '07. IEEE Workshop on*, pages 8–8, Feb 2007.

[17] H. Fassold. Computer vision on the gpu: Tools, algorithms and frameworks. In *2016 IEEE 20th Jubilee International Conference on Intelligent Engineering Systems (INES)*, pages 245–250, June 2016.

[18] T. Limna and P. Tandayya. Video surveillance as a service cost estimation and pricing model. In *2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 174–179, July 2015.

[19] Andrea Prati, Roberto Vezzani, Michele Fornaciari, and Rita Cucchiara. *Intelligent Video Surveillance as a Service*, pages 1–16. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[20] N. Baaziz, N. Lolo, O. Padilla, and F. Petngang. Security and privacy protection for automated video surveillance. In *2007 IEEE International Symposium on Signal Processing and Information Technology*, pages 17–22, Dec 2007.

[21] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. One-shot video object segmentation. *CoRR*, abs/1611.05198, 2016.

[22] Shanpreet Kaur. Background subtraction in video surveillance. Master's thesis, University of Windsor, 4 2017. Electronic Theses and Dissertations. 5944.

[23] Seungwon Lee, Nahyun Kim, Kyungwon Jeong, Inho Paek, Hyunki Hong, and Joonki Paik. Multiple moving object segmentation using motion orientation histogram in adaptively partitioned blocks for high-resolution video surveillance systems. *Optik - International Journal for Light and Electron Optics*, 126(19):2063 – 2069, 2015.

[24] Lun Zhang, Stan Z. Li, Xiao-Tong Yuan, and Shiming Xiang. Real-time object classification in video surveillance based on appearance learning. *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[25] G. Berdugo, O. Soceanu, Y. Moshe, D. Rudoy, and I. Dvir. Object reidentification in real world scenarios across multiple non-overlapping cameras. In *2010 18th European Signal Processing Conference*, pages 1806–1810, Aug 2010.

[26] O. P. Popoola and K. Wang. Video-based abnormal human behavior recognition: A review. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):865–878, Nov 2012.

[27] M. R. Islam, N. I. Shahid, D. T. ul Karim, A. A. Mamun, and M. K. Rhaman. An efficient algorithm for detecting traffic congestion and a framework for smart traffic control system. In *2016 18th Int. Conf. on Advanced Communication Technology (ICACT)*, pages 1–1, Jan 2016.

[28] Bulent Tavli, Kemal Bicakci, Ruken Zilan, and Jose M. Barcelo-Ordinas. A survey of visual sensor network platforms. *Multimedia Tools Appl.*, 60(3):689–726, October 2012.

[29] B. Dieber, J. Simonjan, L. Esterle, B. Rinner, G. Nebehay, R. Pflugfelder, and G. J. Fernandez. Ella: Middleware for multi-camera surveillance in heterogeneous visual sensor networks. In *2013 Seventh International Conference on Distributed Smart Cameras (ICDSC)*, pages 1–6, Oct 2013.

[30] Henry Detmold, Anthony Dick, Katrina Falkner, David S. Munro, Anton van den Hengel, and Ron Morrison. Middleware for video surveillance networks. In *Proceedings of the International Workshop on Middleware for Sensor Networks*, MidSens ’06, pages 31–36, New York, NY, USA, 2006. ACM.

[31] Fortinet Inc. Understanding IP Surveillance Camera Bandwidth. Technical report, 2017.

[32] A. D. Bagdanov, M. Bertini, A. Del Bimbo, and L. Seidenari. Adaptive video compression for video surveillance applications. In *2011 IEEE International Symposium on Multimedia*, pages 190–197, Dec 2011.

[33] F. Xianping and L. Dequn. A new video compression method for surveillance network. In *2007 IFIP International Conference on Network and Parallel Computing Workshops (NPC 2007)*, pages 869–872, Sept 2007.

[34] G. R. Bradski and J. Davis. Motion segmentation and pose recognition with motion history gradients. In *Proceedings Fifth IEEE Workshop on Applications of Computer Vision*, pages 238–244, 2000.

[35] Gary R. Bradski. Real time face and object tracking as a component of a perceptual user interface. In *Proceedings of the 4th IEEE Workshop on Applications of Computer Vision (WACV’98)*, WACV ’98, pages 214–, Washington, DC, USA, 1998. IEEE Computer Society.

[36] João F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *CoRR*, abs/1404.7584, 2014.

[37] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346 – 359, 2008. Similarity Matching in Computer Vision and Multimedia.

[38] Davide Baltieri, Roberto Vezzani, and Rita Cucchiara. 3dpes: 3d people dataset for surveillance and forensics. In *Proceedings of the 1st International ACM Workshop on Multimedia access to 3D Human Objects*, pages 59–64, Scottsdale, Arizona, USA, November 2011.