



# Computer Vision on the Edge to Reduce Network Bandwidth Consumption and Computing Resources in Multi-view 3D Industrial Inspection without Hidden Surfaces

David Millán Escrivá<sup>1</sup> · Javier Tendillo Ruiz<sup>1</sup> · Pau Garrigues Carbó<sup>1</sup> · Andrés Martín Larroza Santacruz<sup>1</sup> · Guillermo Amat Gomariz<sup>1</sup> · Javier Perez Soler<sup>1</sup> · Jose Luis Guardiola<sup>1</sup> · Juan-Carlos Perez-Cortes<sup>1</sup>

Received: 21 April 2021 / Revised: 13 January 2023 / Accepted: 16 January 2023 / Published online: 14 February 2023  
© The Author(s) 2023

## Abstract

Industrial inspection industry requires high precision, fast and reliable systems, where images play a central role. These systems are composed by several hardware and also cyber-physical components where complexity increases when multiple heterogeneous sensor inputs are combined. Our 3D industrial inspection scanner is able to reconstruct complete objects without occlusion with use of multiple sensors and actuators using a complex software architecture. Our system allows increasing the throughput by removing the bottleneck network issue, decreasing network data transfer using a new edge systems architecture that segments and optimizes image transferring. Also, this work presents the results of applying technology developed during the FitOptiVis European ECSEL project. FitOptiVis will provide a reference architecture supporting composability built on suitable component abstractions and embedded sensing, actuation and processing devices adhering to those abstractions. The reference architecture will support design portability, on-line multi-objective quality and resource management and run-time adaptation guaranteeing system constraints and requirements based on platform virtualization. The FitOptiVis project will be applied to design the new architecture of the new edge components and develop the runtime system monitoring.

**Keywords** Industrial inspection · 3D scanner · Edge computing · Computer vision · Image processing · FitOptiVis

## 1 Introduction

Industrial inspection is an important field which requires high precision, fast, and reliable systems. 3D metrology equipment is usually located in a different place than the production floor. Not all the production is usually inspected and requires skilled human labour to perform full surface checks. This process is time-consuming and the production costs increase requiring a specialized person to measure and interpret the results obtained. While manufacturing technologies improve, Computer-Aided Designed (CAD) objects become more complex, and the economic margins at the same time, decrease due to global competition.

Complex products are often composed of many parts that must be assembled, and many measurements must be performed to check all the critical points to assure that the final product has minimal deviations that do not affect to the product quality. The inspection system must be placed in line to obtain 100% inspection coverage automatically with minimal human interaction.

Nowadays, the typical inspection systems are 2D or 2.5D, which are based on frontal, lateral or overhead views, leaving hidden regions that cannot be analyzed. To acquire more views or larger portions of the object, it is usually required to manipulate the objects during long processes with sequential operations using different techniques like stereoscopy, structured lights or laser beams [1–3]. To avoid manipulation of objects, robot arms can be used to move objects in front of sensors to capture all object's sides such as Fei et al. [4] or Brosed et al. [5]. Obtaining 3D reconstruction from multiple views without manipulation is possible as Perez-Cortes et al. [6] show.

---

✉ David Millán Escrivá  
dmillan@iti.es

<sup>1</sup> Instituto Tecnológico de Informática, Valencia, Spain

All of the acquisition techniques mentioned before require complex architectures and component designs, where multiple hardware and software elements are involved and interconnected using video-processing pipelines. Also, these techniques require advanced image processing that are computing-intensive, whereas low latency and high throughput are required for adequate behaviour. Multiple sensors, actuators, boards, etc are involved. Designing and creating a good reference architecture composed of a set of component abstractions is the key to building a stable and scalable system, as FitOptiVis [7] explain as part of the consortium of the FitOptiVis project.

The experiments and results presented in this work use a 3D acquisition system from Perez-Cortes et al. [6] using a ZeroGravity 3D (ZG3D) device [8] as the one shown in Fig. 1 in combination with concepts introduced in the FitOptiVis project [7] to assist in the design of the new components and architecture using a component language and toolset for quality and resource management (QRML) Berg et al. [9]. QRML enables specifying heterogeneous hardware/software systems and their composition and configurations conveniently. The QRML toolset offers language support, visualizations, documentation generation, template-code generation, and constraint-solving support that helps to design and develop new Cyber-physical complex systems. In addition, a monitoring system has been designed using the open-source data visualization and processing framework called IVIS by L. Bulej et al. [10] in order to provide a performance visualization of the previously mentioned device.



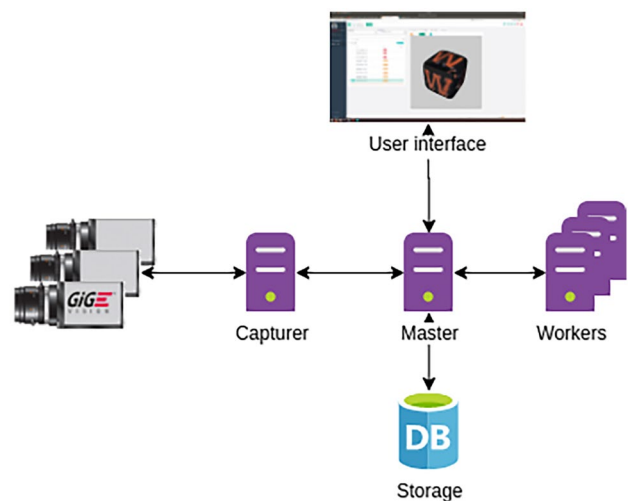
**Figure 1** Industrial inspection system Zero Gravity 3D (ZG3D).

The work is structured as follows. Section 2 describes the current state of the ZG3D industrial inspection system and the motivation of this work to avoid the current ZG3D limitation. Section 3 describes the new proposed ZG3D architecture, component design and monitoring components. Experiments and results after new components and architecture proposed are shown in Section 4. Finally, conclusions and future work are presented in Section 5.

## 2 Motivation

This work starts from ZG3D patented device [8] that is an industrial in-line 3D inspection system. ZG3D employs a single-shot multi-camera acquisition system, using 16 synchronized cameras distributed in the faces of a polyhedron. The object to scan falls through the ZG3D structure and, when it is in its center, a synchronized capture is taken. The capture is sent to be processed, compute the object metrics and generate results, based on which the application automatically decides if the object is flawed or passes the checks. A human inspector can also visualize the progress, 3D reconstructions and measurements of each object processed.

The entire process is implemented in a well-defined architecture, that can be seen in Fig. 2, composed by a capturer, master, one or more workers, database storage and a web interface. The capturer manager is responsible to communicate and synchronize the 16 cameras which are serving a single capture at the same time. The master component manages and orchestrates all devices and software in the architecture. Workers are responsible to process all the captured images and extract surfaces and metrics. The database stores all assets and results from workers and capturer. Finally the web service provides a user interface to control the process and show the results.



**Figure 2** Architecture Zero Gravity 3D.

The 16 cameras with shutter sensors have a resolution of 2448 x 2048 pixels, 3.45  $\mu\text{m}$  RGB square pixels, GigeE Vision interface protocol, lenses with a focal length of 50mm. The diaphragms are set to an aperture of F16 to obtain an adequate depth of field. This configuration allows capturing a maximum object dimension to be inspected of 30mm with a measurement precision of less than 0.02mm. The cameras are connected to a 10Gbps Ethernet switch, each of them is connected to a 1Gbps Ethernet link, and two of the 10Gbps links are connected to the computer system composed of a variable number of high-performance computer nodes. The computation time depends on the complexity of the object, the number of measurements and precision required, and is between 1 and 10s. Many objects can be processed in parallel to increase the throughput of the system. The specifications of the prototype used is presented in the Table 1.

To increase the number of objects processed we need to improve the latency and/or throughput. Latency is more complex to reduce due the limitations related to actuators, sensors, and other hardware involved. The throughput can be improved using several cores and/or computers in a cluster to reconstruct and process the objects.

Actually, the network is the main limitation to increase throughput. Each capture, composed by 16 images of 2048 x 2448 pixels with 3 channels and 8 bits per channel, requires 1.925 Gbits to transport the capture from the 16 cameras to the workers. The cameras use the UDP protocol which does not ensure that the image will be transferred correctly. Although Gige provides some tools to recover incorrectly transmitted packets, we have to ensure that a bottleneck in the network will not exist by increasing transmission rates. Using a 10Gbps Ethernet switch we can only send 5 captures per second. In the laboratory, we tested that we can only send without any packet loss 4 captures per second, since other ZG3D processes are also using the network to communicate.

This work will focus on how to solve the above mentioned bottleneck problem in our network to increase the throughput and minimize the network latency by using edge computing near the cameras. A new architecture is

proposed following the aim of the FitOptiVis project to reduce power-consumption while increasing performance moving the computation from the cloud to the edge. In this use case, computation has been moved from the centralized computation unit (cloud) to distributed small components near the cameras (edge) reducing latency, saving bandwidth thus increasing performance.

### 3 Architecture, Components Design and Monitoring Runtime

ZG3D has a complex architecture where different hardware pieces and software components are interconnected, as we can see in Fig. 3. Images play the main role in the architecture and travel through every component of it. The new architecture has to solve the network bottleneck that occurs when the 16 images per capture, with a resolution of 2448x2048 are simultaneously sent to the capturer manager to process them. Before this proposed solution the images were sent to the capturer with a small delay between images to avoid this issue, but this approach decreases the throughput or number of captures per second. Analyzing the ZG3D topology, we can observe that the 16 cameras are distributed conveniently on the faces of a polyhedron pointing to the center of polyhedron, and the largest volume that can be captured in the small prototype is a 30mm diameter object as mentioned in Perez-Cortes et al. [6]. This maximum object size mentioned does not fill the entire image. Considering the camera focal length, distance to the object and the object largest size, the maximum object size projected in the camera sensor is 6mm. The square pixel size of the cameras used is 3.45  $\mu\text{m}$  then the maximum number of pixels of an object in an image is 1740pixels. Therefore, the largest possible object covers the 51% of a single image. If we remove the non-relevant part of the image and don't send it over the network we can spare at least 49% if network bandwidth at each image transfer.

The proposed solution removes the non-relevant area of the image by cropping the detected object and thus reducing network band with usage before sending images from a camera to the capturer. We use a distributed system, see Fig. 4, where each camera is directly attached to an edge device to process and resend only the relevant part of an image to the capturer. In order to do this, each edge device requires two network interfaces, one connected to the camera and another one connected to the network switch. With this new proposal, a new architecture is required to include this new edge component.

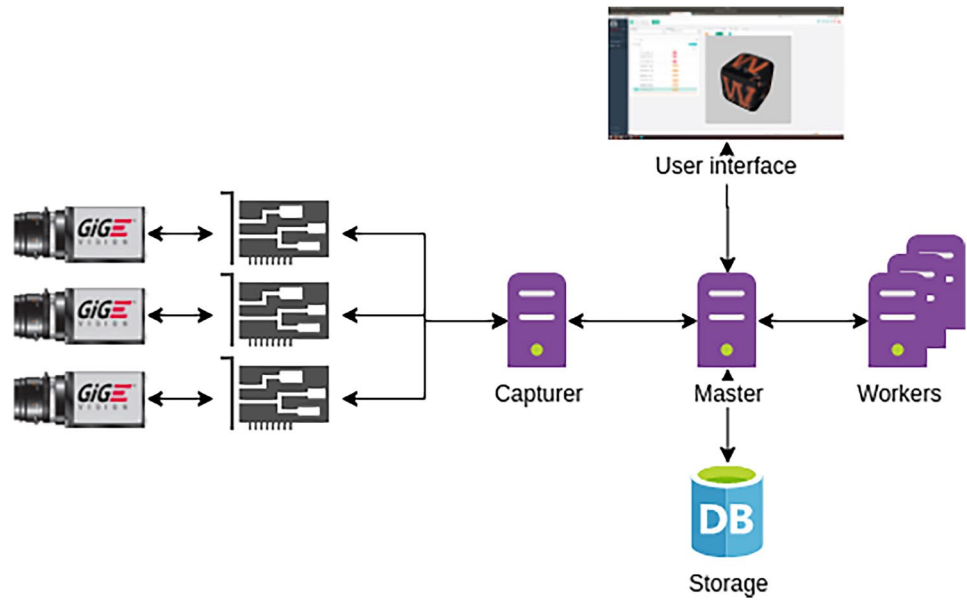
#### 3.1 Architecture and Components Definitions

The new architecture requires high optimization and fault-tolerant hardware/software. To achieve this goal, we

**Table 1** ZG3D prototype used in the experiments.

Component	Specification
Num. Cameras	16
Sensors	Sony IMX264 5.1Mp
Sensor size	2448 x 2048
Max object size	30mm
Resolution	1pixel = 20 $\mu\text{m}$
Throughput	4 captures/s
Latency	1-10s

**Figure 3** New proposed architecture Zero Gravity 3D.

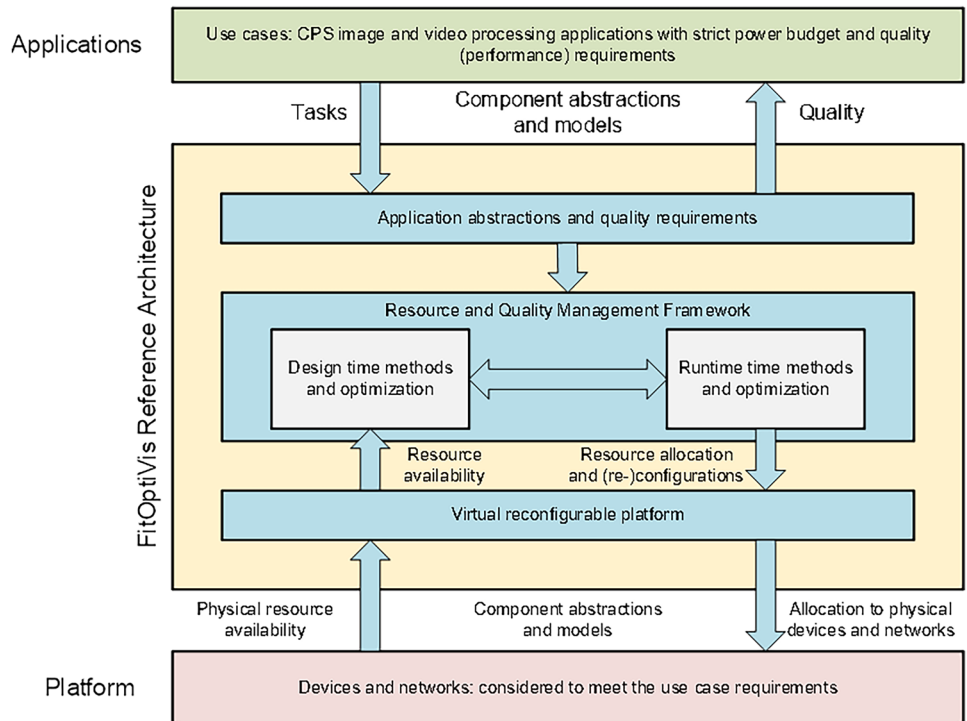


designed the components involved in solving the bottleneck based on the principles of the FitOptiVis project [7]. The FitOptiVis project defines a reference architecture that captures the diversity of platforms, resources, applications, methods, tools and objectives of common projects of image or video pipelines for cyber-physical systems (CPS) on a heterogeneous network. The reference architecture defined by FitOptiVis will be provided as a template of components abstraction on top of a virtual platform offering runtime optimization support for quality and resource management.

The FitOptiVis reference architecture, the yellow box in Fig. 4, takes in consideration a concrete platform and application (red and green boxes) and consists in abstract applications that are considered as a collection of tasks and are reconfigurable depending on the different platform that it is running on, and a virtual platform that shares with applications the resources and capabilities of the attached devices and hardware.

The abstract applications and the abstract virtual platforms are brought together by a resource and quality

**Figure 4** Fitoptivis Architecture.



management framework. This framework is responsible for finding feasible and optimal combinations of applications and platforms as well as optimal operating set points for the components.

Each component is defined with one input and one output, requires a budget and provides a budget, quality and parameters as we can see in the Fig. 5 that is detailed in Hendriks et al. [11]. The components in the architecture have one or more configurations that may be parameterized. A configuration has an input, an output, a required and provided budget and a quality. These parameters allow us to define different working points for each component and runtime configuration.

To design the new architecture and components we use Quality and Resource Management domain-specific Language (QRML) van den Berg et al. [9] that defines domain specific language for QRM, and also provides an accompanying toolset. This toolset allows us to specify a QRML model, model visualization and generate C++ code templates for system and application monitoring.

### 3.2 Use Case Architecture and Components

We designed the new architecture and components using QRML and connect to existing capturer manager with small modifications to the existing architecture. The main component is the whole system called “EdgeSystem” that contains the hardware and application components as we can see on the Figs. 6 and 7.

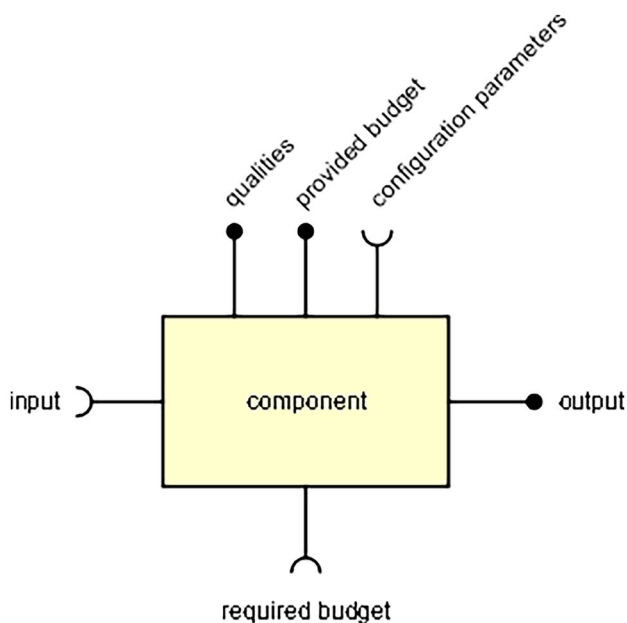


Figure 5 Fitoptivis Component.

The hardware is encapsulated in a virtual component to allow abstract to the specific hardware initially. The virtual hardware component called “VirtualEdgeBoard”, consists of two more components, the board connected to the camera, and the camera itself. The board component defines two possible boards, JetsonBoards and RaspberryPi, and each one exposes their main component qualities, the amount of memory, processing power and if it has GPU capabilities. The camera component defines the required resolution parameter to work in our ZG3D, then if a new hardware is attached to the camera component, the hardware will be discarded if the required resolution requirement is not met.

The application component requires the execution board and the image capturer, both components come from board and camera hardware components, and provide for the application the resources that requires and need as an abstraction to hardware. The application component includes two more components: segmentation and transfer. The segmentation component is the one responsible to retrieve the component budget provided by board and camera components and to segment the image, and the transfer component has the role to send the image and detect any issue in transferring images or bottlenecks in the network to change the segmentation parameter if it’s required.

QRML has different tools to export the QRML language specification to different visual representations, as a components architecture in Fig. 6 and components dependencies in Fig. 7. Also the QRML toolset can export a C++ architecture to provide component abstraction, requirements, dependencies and IVIS connection for monitoring the architecture. These tools allow users focus on developing the main aspects of the application leaving concepts of monitoring and dependencies to the QRML specifications.

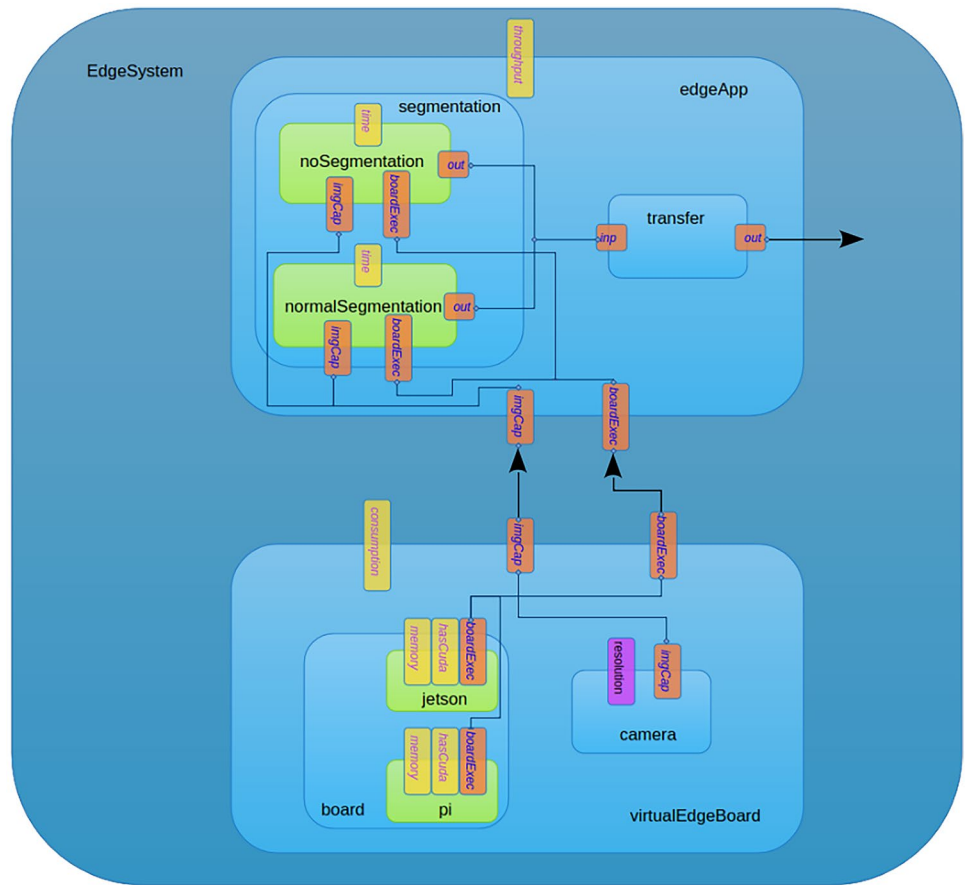
### 3.3 Runtime Monitoring

When a complex architecture with several hardware and sources components are designed, it is important to ensure that all pieces are working properly and don’t raise any defined conflict. A monitoring system can help to meet the constraint and notify to users or architecture when a constraint is not met. In an industrial process a monitoring system is crucial to avoid stopping the production line.

Runtime monitoring can involve monitoring techniques at different levels, from cloud to edge. In turn, for each level a monitoring technique can span different levels from hardware to software level, albeit mainly requiring the synergy of both. This means that the development of monitoring systems in FitOptiVis scenarios, and corresponding system-level services, involves several trade-offs from the architectural point of view.

The 3D Industrial Inspection use case developed by ZG3D requires a monitoring system featuring minimal

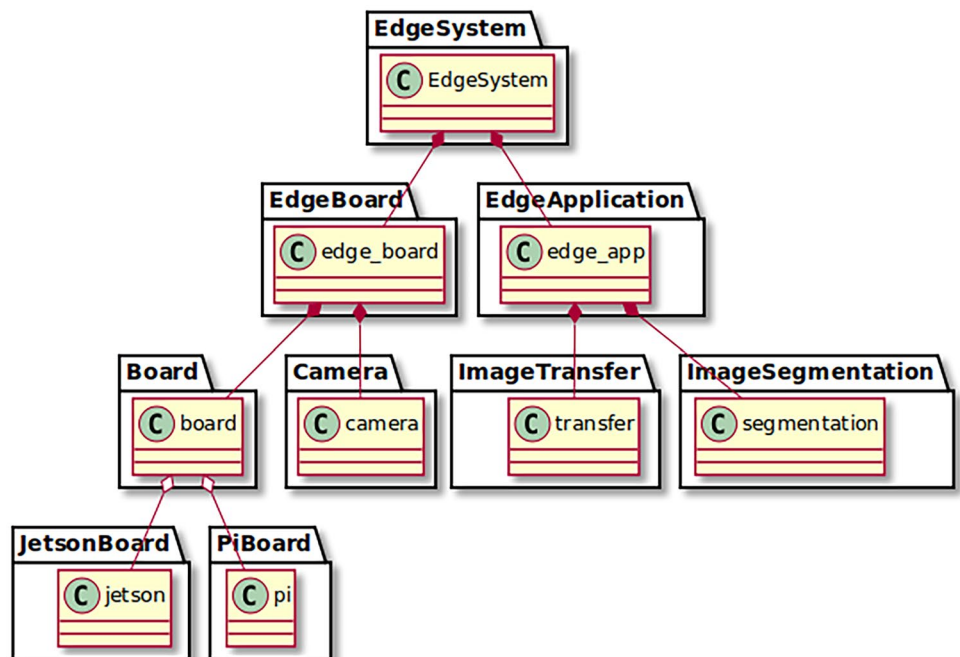
Figure 6 QRML.



intrusiveness and very small bandwidth consumption. The level of intrusiveness depends on the time interval of monitoring events, however, even with a small interval, the intrusiveness should be minimal.

As a general requirement, monitoring must not affect memory and timing performance at the edge. In other words, this process must not delay in any way the tasks performed on the edge capturer. This restriction can be partially avoided

Figure 7 DSL.



dividing the monitoring application into two parts. First, the client-side agent which is responsible for pushing events. This program accomplishes the minimal and non-intrusive requirements. Secondly, the server-side, which can be installed on a different dedicated computer that is in charge of storing the received data and providing the graphical user interface. Regarding the information to be gathered, Zero Gravity 3D collects the following data in order to monitor its state:

- Network bandwidth.
- Throughput, measured as parts per minute.
- CPU load.
- Memory usage.

The new architecture implies the monitoring of sixteen edge computer boards connected to the same number of cameras. In order to manage the state of this hardware, a monitoring system capable of providing realtime monitoring is required. These devices must be monitored in a non-intrusive way and all the data produced must be sent to a central storage to be interpreted.

We employ IVIS as a monitoring platform featuring storage capability and the creation of visualization dashboards from stored data. The monitoring data can be viewed under the IVIS platform in panels created for each edge board as we can see in Fig. 8. To send data to IVIS a RESTful JSON format is employed. In our use case we employ Telegraf [12] to retrieve operating system data. Telegraf is a monitoring agent that collects data

from different sources such as services like databases or web servers and computer built-in sensors. Using Telegraf plugins, data can be filtered, transformed, decorated and finally stored on a file or sent to a database or any other software. Ideally, the data can be transmitted to a server providing storage and computational resources to host an application offering a graphical user interface. This agent software has been installed on all the sixteen edge boards. We created a plugin for sending data to IVIS [10], this platform plays the role of the aforementioned storage server and graphical user interface. The plugin is based on a Telegraf JSON serializer so it is consistent with the Telegraf architecture and provides the serialization of data into a JSON payload which is compatible with IVIS.

### 3.4 Runtime Configuration

FitOptiVis also defines a runtime configuration as a part of its monitoring system as the set of components a system is composed of, their configurations (specified by their parameters and set-points), and their compositions. In fact, such a composition is another component under the FitOptiVis component framework. Subsequently, FitOptiVis defines reconfiguration as an action or a set of actions leading to a change(s) in system configuration. Therefore, based on the impact of actions, we categorize them into three classes as follows:

- Actions adding/removing components. These actions add/remove components to/from the system. These

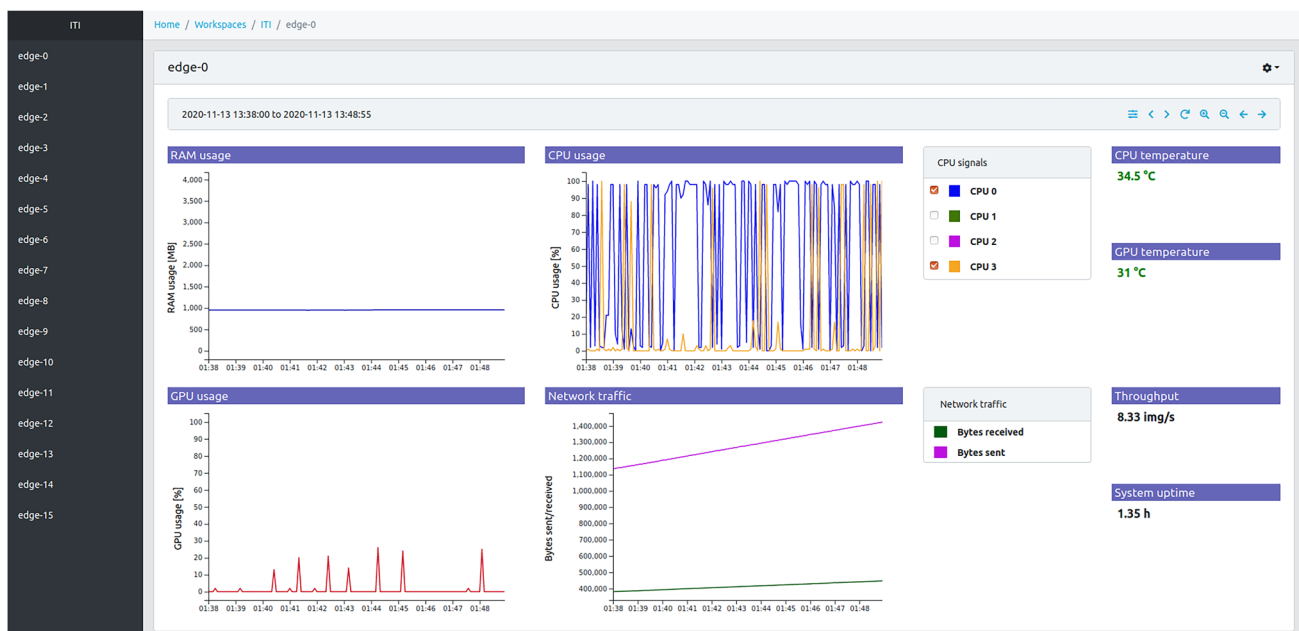


Figure 8 IVIS dashboard.

actions are triggered by users manually, automatically by Quality and Resource Management (QRM) components, or via custom mechanisms from the applications themselves.

- Actions changing component configurations. Configurations of components are defined by set-points their parameters are set at. These actions are triggered either by users, QRM components or applications.
- Actions changing component compositions. Compositions are vertical, horizontal, or free. Vertical compositions have to do with budget connections and are either deployments (application-to-VEP connections) or hosting (EP-to-VEP connections). The actions can establish, modify, or stop connections and are triggered by users QRM components, or applications.

ZG3D can change its configurations in realtime when the throughput decreases or rises over a certain threshold (an action of changing component configuration), and when an edge board is changed to another board with lower or higher capabilities, for example if we change a Jetson Nano board with GPU support to a RaspberryPi without GPU support and less memory (an action of changing component compositions). In both cases this change would lead to a decision in the edge board of performing or not the image segmentation in order to maximize throughput.

## 4 Experiments and Results

A smaller prototype of ZG3D was used as we can see in Fig. 9, where 16 jetson boards are connected directly to each Gige camera and each board connected to the network switch using 1Gbs links. The experiment was done using 2 different objects to capture, a small cube with a printed “M” character and a small sphere to calibrate the system. The objective of experiment was to demonstrate that network bandwidth is reduced after applying a pre-segmentation on the edge. We used the ZG3D software together with the new CamEdge component installed in each Jetson board.

The implemented image segmentation in each CamEdge component was developed on OpenCV with CUDA optimization where the input image is upload to the GPU memory and after that is converted to RGB and a gaussian filter is applied to remove the image noise from the cameras. Finally, an absolute difference using the background and a threshold have been applied. After that, the big area of white pixels is taken as the object in the scene and it is defined as the region of interest that is applied to the original image. This segmentation process takes 90ms on each Jetson Nano edge computing device.

To compare the experiment we use the original ZG3D prototype where cameras are directly connected to the switch



**Figure 9** Prototype with attached Jetson Nano devices.

with an UDP protocol, in this case, the object under inspection is not relevant because no segmentation is being performed (the whole image is sent via the network).

The experiment was executed launching a dice, a capture of which can be seen in Fig. 10. The dice projected into the cameras fills an area of 7% of the image. We use *tcpdump* to capture all the packets sent among the cameras (for original ZG3D) or edge capturer (our prototype) to the capturer. When the capturing process starts we execute *tcpdump* and save all network traces in a *pcap* file to be analyzed. We employ the *Wireshark* software to filter all packets by IP sources exporting only the packets from cameras/edge capturers.

The results of the analysis of the network, shown in Fig. 11 reduce the bandwidth usage per image, decreasing from 15MB/image to 1.2MB/image in our use case. This is a reduction of bandwidth consumption of 92% allowing to increase the throughput and reduce latency of the system. We observe in Fig. 11 that transferring full images requires three times more time than sending the cropped images.

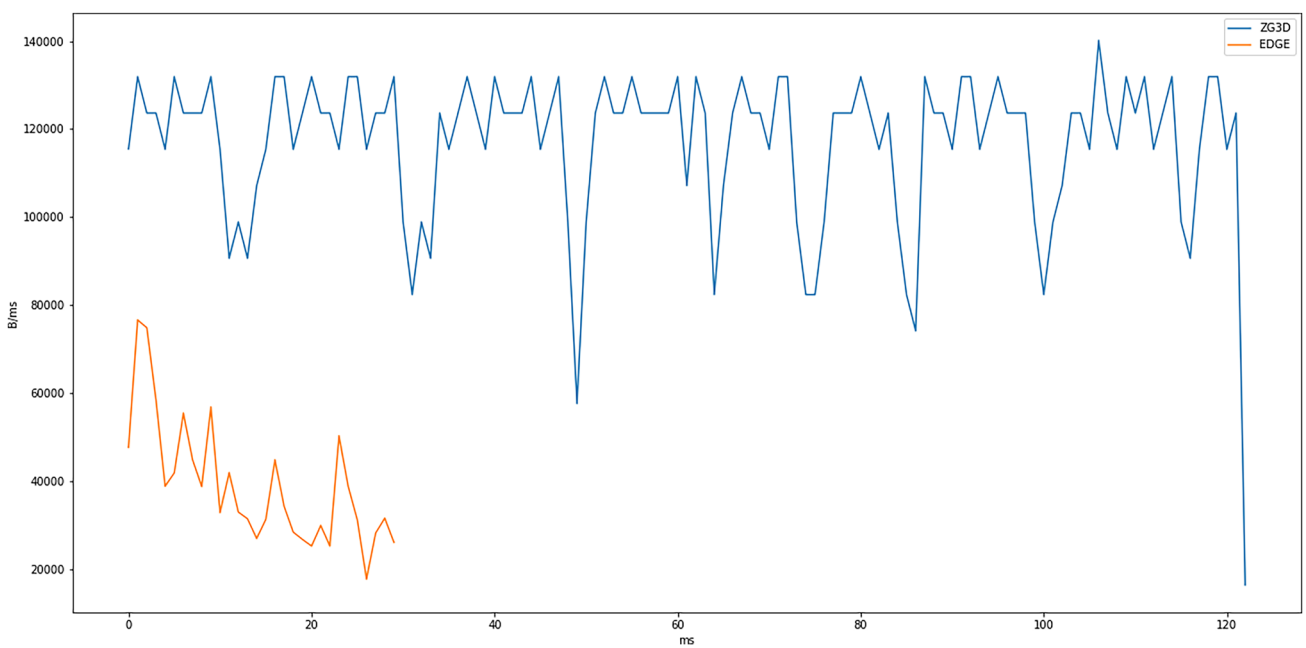


**Figure 10** Object captured by 16 cameras.

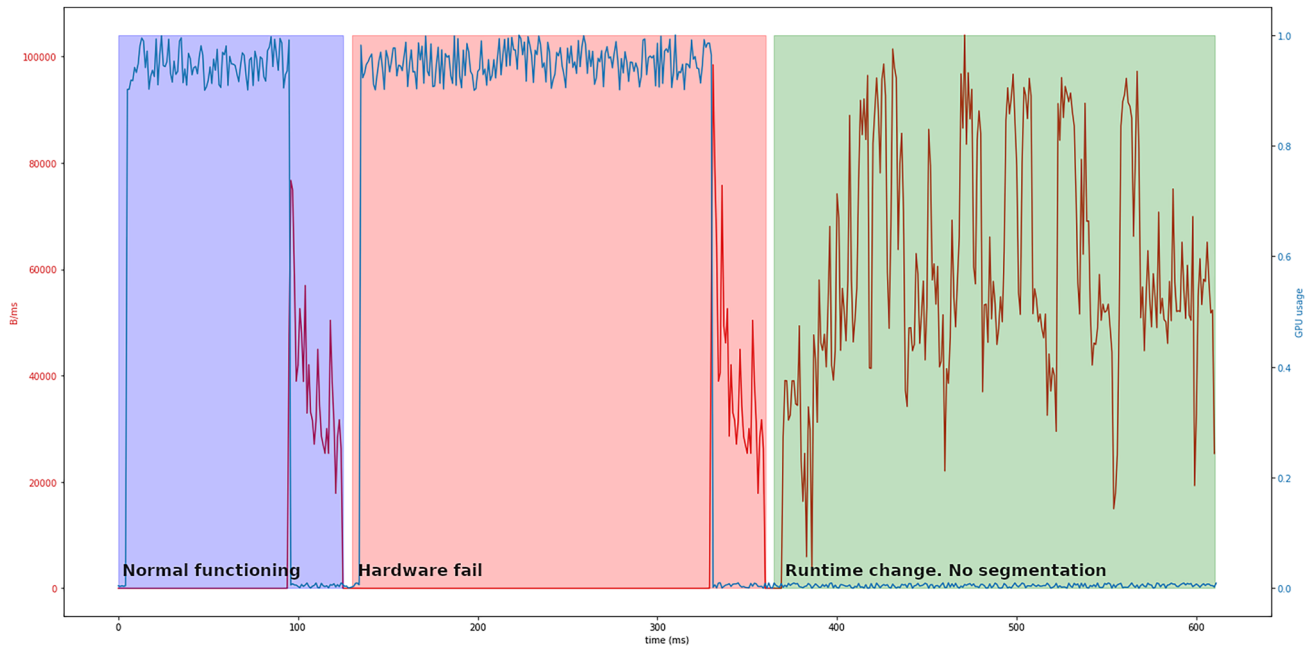


The FitOptiVis runtime monitoring system presented in the use case tries to ensure a minimum throughput on the system. Throughput is monitored inside CamEdge software as high-level monitoring, but we compute the latency time

too, from the receiving trigger of the capture to the last network transfer. The latency is employed as a reference for changing component configurations. In our FitOptiVis use case, we define as a configuration the possibility of



**Figure 11** Network bandwidth Comparison.



**Figure 12** Runtime reconfiguration.

preprocessing or not the image and then send a cropped region of interest (ROIs) or send the whole image. If the latency increases, then the throughput decreases as a consequence. We define a threshold for latency at 150ms. If latency exceeds this threshold, automatically the next capture image is not preprocessed and segmented, removing that computation time and reducing the latency. The processing time can increase due to multiple factors, a common problem can be an overheating of the GPU or a process running out of memory. In our experiments, we forced hardware fails modifying the max power mode of NVidia Jetson Nano that disables 2 CPU cores and deactivates the GPU fan. In Fig. 12 we can see with blue lines the GPU usage and with red lines the bandwidth. In the first blue block, we monitor a lower latency than the threshold defined, where preprocessing consumes less than 90ms. In the next red block, we have hardware fail, where the preprocessing time increases and raises the threshold, then in runtime a reconfiguration of the component is done by disabling the preprocessing and segmentation step. In the

next green block, the new configuration is applied and there is no GPU processing and only whole image transfer is done. With this reconfiguration the system minimum requirements are satisfied.

The reconstruction obtained by the ZG3D software and represented in the interface, shown in Fig. 13, presents an object captured and pre-segmented by edge devices, obtaining a full reconstruction, useful for applying metrological analysis or surface defect detection algorithms without any difference respect to the original ZG3D.

Finally, we compared our solution to a common compression technique. Standard lossy image compression can not be used because of the high accuracy requirements of the application, then we employ a lossless PNG compression. Since the ZG3D cameras cannot perform any image compression, we use the Jetson Nanos as edge computing nodes to compress each image. Depending on the noise present and the characteristics of the object, the time needed to compress each image is  $338 \pm 71,6$ ms on average and the final image size is  $2.2 \pm 0.8$ MB on average.

**Table 2** Experiments results.

Method	Processing time	Transferred data	Packet collision	Total time capture transfer
ZG3D	–	15MB/image	Yes	120ms
Edge segmentation	90ms	$1.2 \pm 0.5$ MB/image	No	$120 \pm 10$ ms
Edge compression	338ms	$2.2 \pm 0.8$ MB/image	No	$388 \pm 15$ ms

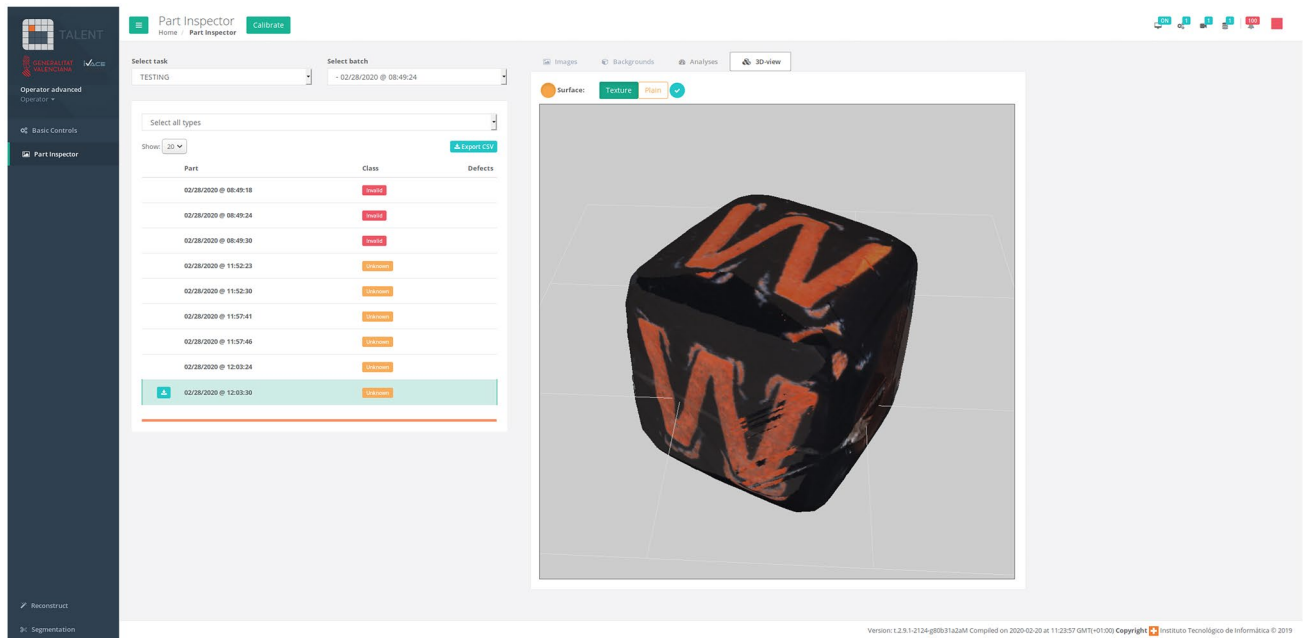


Figure 13 System running.

## 5 Conclusion

We presented a FitOptiVis project use case where a new architecture helped to solve the existing bottleneck; in our test case, the industrial inspection system ZG3D. FitOptiVis provided a framework to create highly optimized and fault-tolerant hardware/software components to attach to the existing ZG3D prototype. The new components designed succeeded in providing the required flexibility, fault-tolerance and monitoring capabilities required for this use case and the system proved to be able to change its configuration in the runtime phase depending on its resources.

We compared our proposal to the original ZG3D system and a compression lossless on the edge (see Table 2). Our system obtains similar results on final transfer time compared with the original system, but our proposal ensures packet transfers and reduces the bandwidth. The compression method requires high computation time and does not reduce the image as much as the segmentation proposal.

As a conclusion, the new architecture designed after analysing the ZG3D issues, helped us to define new procedures to avoid network bottleneck using computing on the edge segmentation, attaching Jetson Nano devices or RaspberryPi to GiGe cameras and pre-processing and pre-segmenting images before sending them to the ZG3D capturer manager reducing at least 39.61% of the bandwidth per object captured and increasing the throughput after solving bottleneck limitations.

**Acknowledgements** This work is part of the FitOptiVis project [7] funded by the ECSEL Joint Undertaking under grant number H2020-ECSEL-2017-2-783162.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Zhang, J., Mai, F., Hung, Y. S., & Chesi, G. (2009). 3d model reconstruction from turntable sequence with multiple-view triangulation. In *International Symposium on Visual Computing* (pp. 470-479). Springer, Berlin, Heidelberg, 2009.
- Fremont, V., & Chellali, R. (2004). Turntable-based 3D object reconstruction. In *IEEE Conference on Cybernetics and Intelligent Systems*, 2004. (Vol. 2, pp. 1277-1282). IEEE.
- Kazó, C., & Hajder, L. (2012). High-quality structured-light scanning of 3D objects using turntable. In *2012 IEEE 3rd International Conference on Cognitive Infocommunications (CogInfoCom)* (pp. 553-557). IEEE.
- Fei, Z., Zhou, X., Gao, X., & Zhang, G. (2017). A flexible 3D laser scanning system using a robotic arm. In *Optical Measurement Systems for Industrial Inspection X* (Vol. 10329, pp. 1190-1195). SPIE.
- Brosed, F. J., Aguilar, J. J., Guillomía, D., & Santolaria, J. (2010). 3D geometrical inspection of complex geometry parts using a novel laser triangulation sensor and a robot. *Sensors*, *11*(1), 90-110.
- Perez-Cortes, J. C., Perez, A. J., Saez-Barona, S., Guardiola, J. L., & Salvador, I. (2018). A System for In-Line 3D Inspection without Hidden Surfaces. *Sensors*, *18*(9), 2993.

7. Al-Ars, Z., Basten, T., Beer, A. D., Geilen, M., Goswami, D. Jinen, P. Kadlec, J., de Alejandro, M. M., Palumbo, F., Peeren, G., Pomante, L., van der Linden, F., Saarinen, J., Sntti, T., Sau, C., & Katuscia Zedda M. (2019). The FitOptiVis ECSEL project: highly efficient distributed embedded image/video processing in cyber-physical systems. In Proceedings of the 16th ACM International Conference on Computing Frontiers (CF '19). Association for Computing Machinery, New York, NY, USA, 333-338.
8. Perez-Cortes, J. -C., & Siez, S. (2014) Barona. Device and method for acquisition and reconstruction of objects, August 19 2014. US Patent 8, 811,722.
9. van den Berg, F., Čamra, V., Hendriks, M., Geilen, M., Hnetyka, P., Manteca, F., ... & Basten, T. (2020, September). QRML: A component language and toolset for quality and resource management. In 2020 Forum for Specification and Design Languages (FDL) (pp. 1-8). IEEE.
10. Bulej, L., Bureš, T., Hnětynka, P., Čamra, V., Siegl, P., & Töpfer, M. (2020, August). IVIS: Highly customizable framework for visualization and processing of IoT data. Charles University, Faculty of Mathematics and Physics, Prague, Czech Republic.
11. Hendriks, M., Geilen, M., Goossens, K., de Jong, R., & Basten, T. (2020). Interface modeling for quality and resource management. arXiv preprint arXiv:2002.08181.
12. Wang, S. (2021). Telegraf Open Source Server Agent InfluxData. [Online]. Available: <https://www.influxdata.com/time-series-platform/telegraf/>. [Accessed: 16-Feb-2021].

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.