Accepted Manuscript version of the article published in *IEEE Transactions on Automation Science and Engineering*, 2022, 19 (3), p. 2171-2186. Available at https://doi.org/10.1109/TASE.2021.3078787

Design and deployment of a generic software for managing industrial vision systems

Núria Banús, Imma Boada, Pau Xiberta, and Pol Toldrà,

***Abstract*—Computer vision systems have become a key element of modern companies. However, its implantation not only requires specialized hardware components but also software to control the correct performance of all these components. The purpose of this article is to present the design and deployment of a software created to manage the different computer vision systems of any company. First, using our experience in mechanization and industrialization of company processes, a general company specification will be proposed as well as the main software requirements that have to be satisfied. Then, a three-level modular design composed of the core, a configuration module and user interfaces with functionalities capable of satisfying the defined requirements will be presented. Special attention will be given to time restrictions and components' synchronization. In addition, the different tests that have been carried out to control the correct performance of the software will be shown. The development process will end with a generic, modular and scalable software able to fit different industrial scenarios by simply modifying a set of input parameters. To illustrate the correct performance of the proposal, the details of its installation in four real companies with different needs will be presented. The proposed work has a practical use in industry and it also provides a thorough description of the main components involved in computer vision systems of real company environments and how to manage them.**

***Note to Practitioners*—The aim of the paper is to provide a software solution to manage the computer vision systems of a company. The proposed software is simple enough to be controlled by company operators with no need for great expertise on the technical aspects, but rather on the industrial workflow to be controlled. To develop such a software, it has been necessary to know about all the elements involved in the industrial scenarios controlled by computer vision systems as well as their relationships. The paper presents all these elements providing a global and complete view of the problem with special attention to synchronization, which is fundamental to satisfy industrial time restriction requirements. The software is presented from a technical point of view, but also from a practical one, since different examples of its application in real cases are given. These examples illustrate the adaptability of the proposal to different scenarios and also exemplify how the different elements are modified to fit the different situations. In addition, the paper provides a comprehensive overview of real industrial situations, focusing on the most important issues faced during real installations.**

***Index Terms*—Computer Vision for Automation, Computer Vision for Manufacturing, Factory Automation, Software**

I. Introduction

OVER the last few years, computer vision systems have become an essential component of industrial processes [1]–[5]. Moreover, its combination with artificial intelligence strategies [6], [7] and other technological advances has allowed the automatic reproduction of many human procedures leading to more efficient and effective industrial processes and outcomes [8]–[21]. However, bringing a computer vision system to production is difficult since different software and hardware elements, such as sensors, mechanization components, illumination systems and image acquisition devices need to be integrated [22]–[24]. Such a difficulty has lead computer vision systems to be, in most cases, domain specific. Systems are designed considering the main procedures and processes that have to be automated, and specific solutions that meet the needs of the studied scenarios are created. In the food industry [25], for instance, specific approaches have been proposed to classify the meat [26], to recognize food images [27], to automate the segmentation of animals' body parts [28] or to determine the moisture and insoluble impurities content in virgin olive oils [29]. In the automotive industry, some methods have been presented to classify the mechanical fractures on metallic materials [30], or to estimate the mechanical properties of the nodular cast iron [31]. For a survey on the applied methods in this area see [32]. In the pharmaceutical industry, methods to monitor emulsions [33] or to predict the compressive strength of consolidated molecular solids [34] have also been proposed. The construction industry has also taken advantage of automated processes [35], which are distinctively useful to ensure the workers' safety [36], [37]. More examples can be found in the textile industry [38], electronic components manufacturing [39], and many others [40].

Independently of the application area, in the core of computer vision systems there is the software that determines how all the connected components have to proceed to perform the desired industrial tasks. Roughly speaking, this software can be described as an application that configures and controls the components of the production line in order to produce the desired outputs. It is composed of different modules designed to [41]: (i) acquire the information from the process or the products, which requires a control over the production line as well as over the acquisition and illumination devices, among others; (ii) process the collected information via image processing techniques and artificial intelligence strategies in order to extract the relevant information that will determine the actions that have to be carried out, and (iii) return these actions as output to control the components of the production line. Obviously, all these procedures have to be synchronized to ensure that the scenario's time restrictions are satisfied. Since similar situations require quite similar strategies, this software is generally able to fit variants of the same case. For instance, in quality control processes, the computer vision software

N. Banús, I. Boada and P. Xiberta are with the Graphics and Imaging Laboratory from University of Girona, Catalunya.

N. Banús and P.Toldrà are with TAVIL Ind. S.A.U., Girona, Catalunya

determines when and how product images have to be acquired. This image acquisition process will be different depending on the features of the product such as the size and the type of packaging. Similarly, the processing of acquired images will be different depending on the specific quality requirements of the product. For this reason, to support these variants, the computer vision software provides a graphical user interface to adjust its parameters according to the products.

Industry automation requires different computer vision software systems to coexist. Moreover, there is a great interest in the industrial serial production systems with flexible manufacturing capable of producing different types of products with finite production runs on the same production line by equipment adjustments and setups [42]. Therefore, besides the software of the computer vision system, a high-level application is required to manage all the systems and decide at all times how they should proceed. This new software will allow all the systems to be controlled holistically, thus centralizing all the actions and leading to a more efficient management. To tackle this problem, different ready-made solutions are available. However, some companies are interested in the development of in-house solutions to avoid the limitations of proprietary systems. This is the case of our company which is specialized in the construction of machinery and automatic lines for packing, palletizing and handling. It designs, develops and produces customized solutions and turnkey projects worldwide. To meet the customers' customized demands in a shorter time, an in-house product is considered to be suitable as it will require less maintenance as well as a lower reconfiguration cost. The development of such a software is not an easy task, considering that it will be responsible for all computer vision systems, the control of sensors and devices, and the processing of related data following the restrictions imposed by the production line. In addition, the proposed software will have to be scalable, reusable, flexible and extensible, among others, to fit the changing requirements of the customers with a minimal impact on the source code.

Focusing on the development of this software, the goal of this paper is several-fold. First, to present the study that has been carried out to identify the main components involved in the development of a software capable of managing the different computer vision systems of a company. To unify the possible use cases that have to be supported, the specification of all these components is required. Second, driven by the proposed specification, to describe the developed software with a detailed description of its modules and with special interest in communication and synchronization issues. Third, to explain the different experiments that have been carried out to evaluate the efficiency and adaptability of the proposed software to different real scenarios with different needs.

The main contributions of the paper can be summarized as follows: (1) The identification and contextualization in a real industrial scenario of the main elements involved in the development of a software to manage different computer vision systems as well as their relationships, with special attention to communication and synchronization details; (2) The design and implementation details of a generic, modular and scalable software capable of controlling different computer vision systems and fit different industrial scenarios by simply modifying a set of input parameters; (3) The evaluation of the managing software in different real scenarios by considering different use cases.

Note that the proposed software would be a key element of a more general framework for adaptive industrial automation where input parameters are automatically fixed, thus reducing user interaction.

II. Methods and materials

To create the software, our development has been driven by a general analysis and requirements gathering, a software design process, coding and testing, and software deployment. All these steps are described in the next sections.

A. General analysis and requirements gathering

In the first phase of the project, we took advantage of our experience in the field of mechanization and industrialization of company processes with transport lines, supply processes and product management, among others. With the idea of developing a flexible, configurable and interoperable software capable of adapting to as many companies as possible, we proposed a set of terms to specify a general company situation that has to be mechanized and controlled via computer vision techniques.

1) Company specification: The terms introduced to describe company needs are illustrated in Figure 1 and presented below. The *context* is the space of the company that has to be mechanized and controlled via computer vision techniques. The context is composed of one or more *environments*, each one with its own components such as conveyor belts and robotic arms, as well as with computer vision elements such as *cameras* and lights, and other elements such as sensors and encoders to perform the different actions. Each environment can support different *scenarios* which depend, for instance, on the flow of products that they have to deal with. In the example, environment 4 supports two scenarios, scenario 1 and scenario 2, that differ in the number of products per box. To determine how the scenario has to proceed, we introduce the *project* term. A project contains the specifications that one environment requires to support one scenario. The computer vision system processes the cameras' images using a computer (*PC*) and applying image processing methods referred to as *analysis*. The analysis has input/output parameters to represent the required/returned information, respectively. The scenarios are governed by a Programmable Logic Controller (*PLC*) that determines the different actions that have to be carried out for the *communication* of all components.

2) Software requirements: Once all information related to the company specification has been identified, we define the software requirements. In particular, our software has to:

- Provide functionalities to define the context, the environments, the cameras and the communication from scratch and with minimal effort. Different camera models have to be supported with different features, different management parameters and different interfaces.

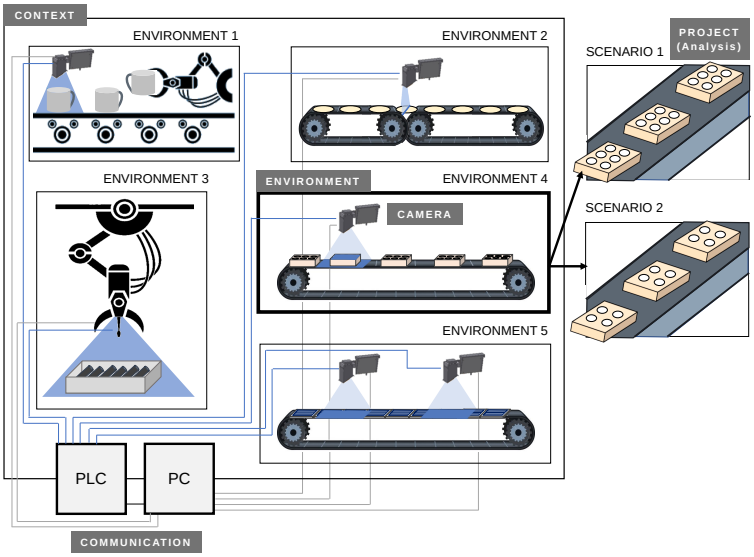


Fig. 1: Terms introduced to describe the main elements of the company involved in its mechanization and computer vision control.

- Support the definition of any context with no restrictions on the number of environments, scenarios, cameras and communication.
- Be independent of communication protocols, information load resources or computer vision technologies to ensure its adaptation to any scenario.
- Support the project definition and the change of projects at running time to fit changing scenarios.
- Support the modification of camera configuration parameters in real time.
- Support real-time and periodic input and output communication not only with the PLC but also with other elements that may be considered necessary for cooperative work.
- Support the selection of projects from the PC or the PLC in real time ensuring consistency between both components.
- Provide real-time information of any scenario.
- Support an offline working mode to combine real production with the development and testing of new functionalities with no impact on production. This requirement involves the possibility to (i) process images already captured by cameras and (ii) save captured images.
- Optimize computation time to ensure an efficient and effective scenario working mode. The application must achieve the minimum run time in order to optimally work.
- Control errors and camera recovering protocols to recuperate from fails in minimum time.

Note that in most requirements there is a reference to real time, and the real time depends directly on the response time, which includes the computation time. Obviously, the response time has to be optimal, i.e., the minimum, since high-speed operations are demanded in order to increase the production rates. A real-time system is one whose logical correctness is based both on the correctness of the outputs and their timeliness [43]. To satisfy these requirements, different methods have been proposed to develop real-time software applications which accomplish a sequence of processes with temporal restrictions on dynamic environments [44]. In addition, as this demand also depends on the hardware, high-speed vision systems have been also proposed [45], [46]. In our context, to deal with all these issues, and inspired by software development methods such as Jackson System Development (JSD), Real-Time Structured Analysis and Design (RTSAD), Design Approach for Real-Time Systems (DARTS) and Yourdon Structured Method (YSM) [44], [47], a software design has been considered where concurrency, real time and object-oriented programming are essential. As final considerations, the application has to be responsive, multi-user, with support to different languages, and modular to support the integration of new functionalities in an easy way.

B. Software design

To design the software, we grouped all our requirements focusing on the working pipeline and considering the expert and the basic user profiles. The expert has to determine the working mode for all the scenarios of the company, which requires the context specification with the definition of environments, cameras, communication technologies and projects. Once the working pipeline has been defined, the basic user can control it by determining the projects that have to be executed or by modifying the parameters of the projects. Taking into account all these considerations, the three-level architecture illustrated in Figure 2 and described below is proposed.

1) Core: From the bottom to the top, the first level is the core of the software, which is composed of three modules: the environment module, the camera module and the communication module.

The *Environment Module* has two components: (i) the *Environment Specification*, that contains an identifier and a list of its cameras, and (ii) the *Execution Mode*, that controls the projects of the environment using the identifier and the name

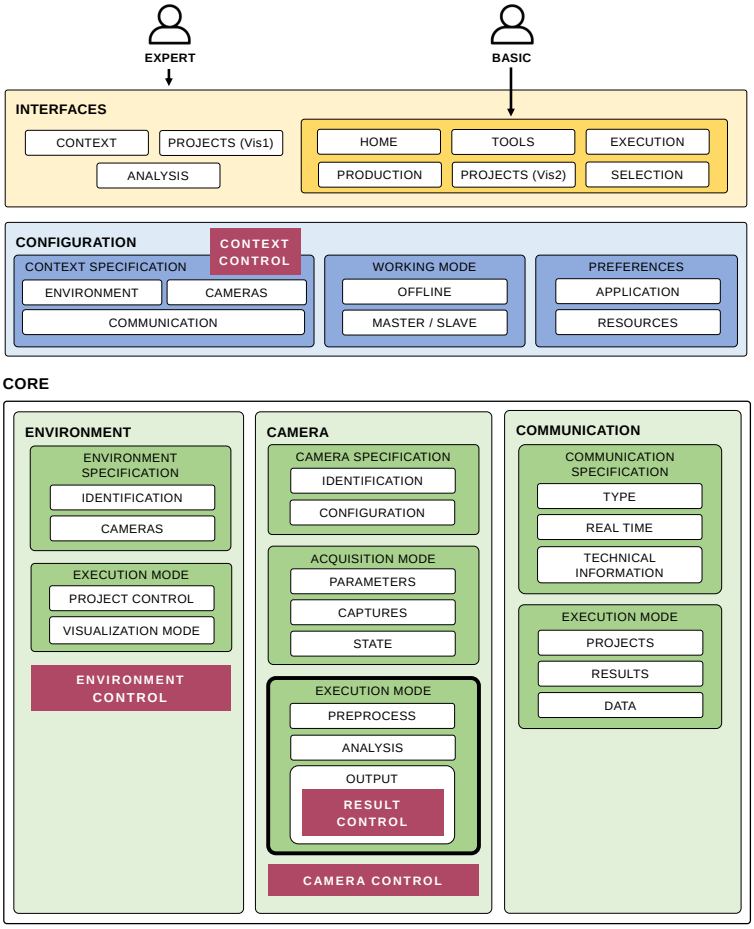


Fig. 2: The three-level architecture of the proposed software with its main components. The control classes are responsible for the synchronization.

of the current project, if it is active, and also the identifier of the last one. This component also controls the information that will be presented in the *Execution interface* described later.

The *Camera Module* has three components: (i) the *Camera Specification*, that contains the serial number, the name, the model, the configuration file and the camera trigger type (internal or external); (ii) the *Acquisition Mode*, that has the configuration parameters that can be modified in real time for each acquisition or for a specific project, the number of acquisitions the camera has to make for each acquisition order, and the camera state, i.e., on (enable/disable) or off (note that the same camera can proceed in different ways depending on the scenario needs); and (iii) the *Execution Mode*, that includes the pre-processing, the analysis process and the output blocks. The pre-processing block indicates the operations that have to be carried out before the analysis process in special situations, such as when different images are required to evaluate a single element, and when one image has to be divided into a set of images to evaluate different features. The analysis process block defines the image processing algorithms that have to be applied according to the input/output parameters. Three types of parameters are considered: images (either acquired images or the ones obtained after processing), iconic parameters (objects such as shape models or regions, calibration files for robotic arms interaction, neural networks for deep learning strategies, etc., or labels that are used or generated in the processing step to better interpret the obtained results) and control parameters (values required to control the execution or values obtained after the analysis). Finally, the last block of the Execution Mode component is the output, which returns the results after processing and also warnings. These results are represented as a set of states, each one indicating success or failure; in case of failure, the states also indicate the identifier of the errors to access the list with their descriptions. Note that the Execution mode corresponds to the core of the computer vision system, i.e., the one that determines how the system have to proceed according to the requirements set by the industrial application and the results obtained from the image processing techniques and the artificial intelligence strategies. These techniques and strategies are specifically designed for the application and no restrictions are imposed on the type of algorithms to be applied. The module can access different image processing libraries such as Halcon [48], OpenCV [49] or Cognex [50], among others. All of them have the latest state-of-the-art machine vision techniques, such as comprehensive 3D vision, deep-learning models and pattern matching [41]. Therefore, the proposed software makes it possible to easily integrate existing image processing and artificial intelligence

techniques. More details on how these techniques apply to real cases will be presented in Section III.

The *Communication Module* has two components: (i) the *Communication Specification*, that defines the communication type, whether it is real-time or not, and technical information such as Internet Protocol (IP) addresses or port numbers, and (ii) the *Execution Mode*, that contains information about the projects (the relationship between active projects and environments), results (true/false values according to processing, errors, and output control parameters of the last analyzed image to be communicated) and data (information that will be presented in a user interface or input control parameters).

2) Configuration module: The second level of the architecture is the *Configuration Module*, which contains three components: the context specification component, the working mode component and the preferences component.

The *Context Specification* component interacts with the Core to provide the information required to define the specifications of the Environment, the Camera and the Communication Modules.

The *Working Mode* component controls whether the proposed software has to run offline or online and also synchronization details such as the application behaviour (master or slave) to solve possible inconsistencies with the components it communicates to.

Finally, the *Preferences* component provides functionalities to select the language; resources such as directory paths to obtain files, elements to be loaded, camera configurations and procedures to be executed; screen features; and the level of detail of register files, among others.

3) User interface: The third level of the architecture maintains all the user interfaces that give access to the software functionalities. These have been grouped in seven interfaces named Context, Projects, Analysis, Production, Selection, Execution, and Tools, which can be accessed via the main interface named *Home*.

The *Context interface* is used to create a context or to modify the created one. Generally, the creation process is carried out only once and the entered information is registered in an XML file that is rarely changed.

The *Projects interface* allows the user to create or modify projects. Entered information is also stored in an XML file. This interface provides two visualization modes, the first ($Vis1$) being used for the expert to create and modify projects that are not running, and the second ($Vis2$) being used to modify in real time the active project of a current scenario introducing changes to the input control parameters, the configuration parameters of the camera or the acceptance or rejection of a product in case of errors.

The *Analysis interface* allows the user to create or modify the analysis specification, i.e., the name of the procedure and the attributes of input/output parameters such as the title of output images, the visual features of each iconic parameter of an image, the description, the units and, when applicable, the range of acceptance of each output control parameter. As in previous interfaces, entered information is registered in the corresponding XML file.

The *Production interface* allows the user to access, through an external application, all data from a context to evaluate the performance. Data is provided by our software as well as by other external sources related to other production processes and controls.

The *Selection interface* allows the user to see all the possible projects of each Environment Module as well as the active ones. It also allows to upload new projects from the application to modify the scenario working mode by changing active projects from the Execution Module and communicating changes to related external components.

The *Execution interface* allows the user to see in real time all the context information according to the specifications defined in the Visualization Mode of the Environment Module of the Core. It shows the environments, and the active project of each environment as well as all its active cameras. In addition, for each camera, it displays the outputs of the last performed analysis presenting the output images (identified by a title), the iconic parameters (displayed with selected colors and visual features) and the control parameters with their description, the corresponding units, and the green or red color, when applicable, to determine whether they are within the acceptable range. This screen also displays accumulated counters and current information of correct and incorrect results and errors. The data selected in the Execution Mode of the Communication Module will also be displayed. This interface can directly access to the $Vis2$ screen of the Project interface. It can also provide information of the communication states. In addition, the Execution interface provides functionalities to detect errors and warnings, thus avoiding possible problems. These errors and warnings can be explored in detail to act accordingly.

The *Tools interface* provides functionalities to access the information created or required during the project. These functionalities allow the end user to perform different actions without the support of an expert.

It should be noted that, although all entered information is registered in XML files that are transferred to the modules, other transference approaches are supported. The system has been designed in such a way that applying a new approach requires only the implementation of methods to access the data and transfer it to the proper modules without effects on previous developments or execution modes.

The access to the functionalities provided by these interfaces depends on user privileges. These are defined via a Comma-Separated Values (CSV) file where for each user there is a description, an access code and the type of permission to the interfaces. As shown in Figure 2, the expert can access all the interfaces while the basic user can only access Home, Execution, Selection, Production, $Vis2$ screen of Projects, and Tools.

C. Control classes

In a real company, each environment requires its own sequence of events to control how all the components have to proceed while satisfying the imposed time restrictions [42], [51]. These events range from a simple camera trigger to a set

of complex actions to control processing, motion, sorting, light sources, encoders, etc. In our software, the four classes that control synchronization and time restrictions are the *Context Control*, the *Environment Control*, the *Camera Control* and the *Results Control*. To describe them, we will first consider the main components to be communicated, then the sequence of actions involved in time restrictions, and finally the designed classes to support them [44].

In Figure 3, a basic company configuration with a single PC, a single PLC and different cameras distributed in the different context's environments is illustrated. The PLC, the PC and the camera are the components that define the synchronization. The PLC centralizes the synchronization providing real-time executions and controlling, among others: (i) the conveyor belts and reject systems via the Servo Driver and using real-time communication protocols via Ethernet; (ii) the robotized arms through their controllers and communication bus; (iii) cameras, lights, and sensors via wired input/output; (iv) the PC using a secure communication protocol; and (v) the external applications using a suitable protocol. The PC and the cameras are connected via a high-performance industrial camera interface supporting different protocols such as GigE Vision [52], USB3 Vision [53], and GenICam [54], among others. The PC processes the acquired images in the acquired order and returns the information to the PLC in the proper order. No errors, network saturation, multiplexing or connection-oriented communication has to be guaranteed. Protocols such as Transmission Control Protocol (TCP), Process Visualization Interface by B&R (PVI) [55], Open Platform Communications Unified Architecture (OPC UA), etc. have to be supported to fit different companies' needs. In addition to being connected to the PLC and the PC, the camera can be wired to the encoder for a correct acquisition, which is synchronized with the movement of context components such as conveyor belts. Different configurations of the camera buffer that manages the acquisition sequence have to be supported as well. Moreover, when defining the environment performance this internal camera setup must be taken into account.

In Figure 4, a detailed description of the sequence of actions between the PLC, the PC, and the camera has been represented with a number that indicates the execution order. In addition, the application classes and the workflow to control the context, the environments, the cameras, the results and the images to be shown in the Execution interface are provided. To begin, we will focus on the sequence of actions that are carried out to support a basic company configuration. The process starts when the PLC receives the *Action order*. Then, it sends the *Acquisition order* to the camera, which responds with an *Acquisition done* message when the image has been acquired. The acquisition time depends on the camera model [45], [46] and comprises the time between the trigger action wired from the PLC and the *Acquisition done* message sent from the camera to the PLC. For offline scenarios, this time is not considered. The PLC, via an *Acquisition ready* message, communicates to the PC that an image ready to be processed is available. At this moment, the PC, which may be managing other situations (*Management*), continues its work and also accesses the camera to load the image (*Acquisition loading*), sending an *Acquisition loaded* message to the PLC when loading has finished. Then, it processes the image (*Processing*) and prepares the *Result* that has to be communicated to the PLC (*Result communication*), which responds with the *Result response* message. The time required by the PC to perform all the actions (see Figure 5) depends on the camera acquisition configuration determined by the Acquisition Mode, and also on the information of the pre-processing, the analysis, and the output blocks of the Execution Mode that will determine the management, processing, and result time. While all these time values are relevant for time performance, the last PLC message, *Result response*, is only used to check that everything is working properly and, in case of problems, to indicate to the PC what is failing to proceed accordingly.

As it has been seen, the PLC manages events by determining the actions that have to be carried out in each environment. It has to simultaneously control the acquisitions from the different cameras and efficiently manage the results in real time. Since working time values are very short, repetitive sequences become concurrent and, when there are several environments with several cameras, they become parallel. To support all these sequences, our design relies on four main classes that control the context, the environment, the camera and the results, respectively.

The *Context Control* is a class from the context specification component of the Configuration Module. It is responsible for the context management and it has been implemented to be technology-independent. As shown in Figure 4, it maintains an array of *Environment Control* classes where each environment class maintains an array of *Camera Control* classes defined according to the project control of the Execution Mode component of the Environment Module. This last array is used by the two execution workflows of the Environment Control named *management workflow* and *result workflow*. The management workflow sequentially controls all the environment cameras following the information of the Camera Module, which determines the camera actions such as (i) camera opening and camera configuration (according to the Camera Specification), (ii) iconic input parameters search in load resources (according to the analysis input parameters of Execution Mode), (iii) camera acquisition (according to the Acquisition Mode) and (iv) creation and execution of *processing workflows*. Actions (i) and (ii) are done once for each camera and current active project. The management workflow creates different processing workflows for each acquisition loading. These workflows are defined and automatically adapted according to the Execution Mode component of the Camera Module from the Core. Each processing workflow returns a result that will be processed by the *result workflow* using the *Result Control* class (according to the output of the Execution Mode from the Camera Module). The obtained result will be communicated according to the Communication Module and, in the event that the user requests a visualization, it will be stored in a queue data structure to be shown in the correct order in the Execution interface; otherwise, it will be deleted at the end.

In Figure 6 two examples of the messages between PLC and PC are presented. The first one describes the information communicated from the PC to the PLC for the *Acquisition*

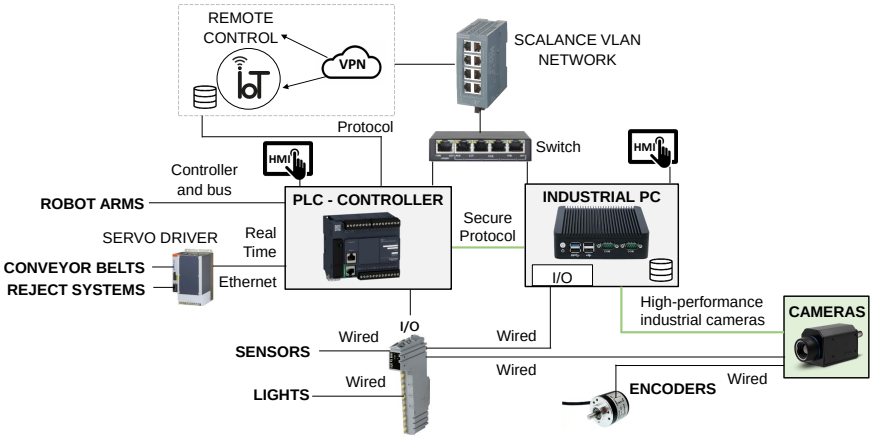


Fig. 3: Basic company configuration with a single PLC, a single PC and different cameras distributed in the different environments.

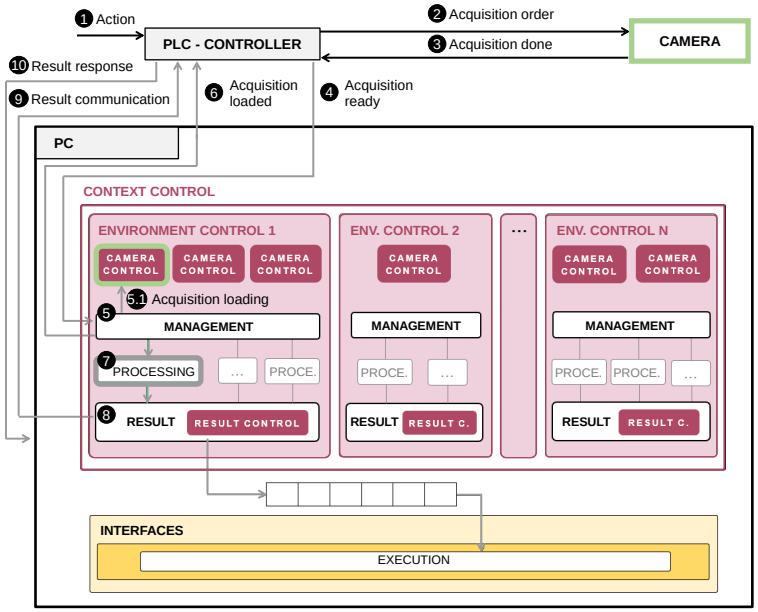


Fig. 4: Description of the sequence of actions between the PLC, the PC, and the camera with a number that indicates the execution order, as well as the application classes and the workflow to control the context, the environments, the cameras, the results and the images to be shown in the Execution interface.

loaded and *Result communication* actions. In the first case, the message sends the identifier of the last loaded acquisition and the acquisition state, and in the second case, it sends the identifier of the last processed acquisition with the prepared result, the state of the internal results, the type errors in case of error, and the output control parameters to be communicated. The second message describes the information communicated from the PLC to the PC for the *Acquisition ready* and *Result response* actions. In the first case, the message sends the identifier of the new *Acquisition ready* and the communication input control parameters, and in the second case, it sends the identifier of the last result communication received and the code of the result state, which can be 0 (timeout), 1 (correct), 2 (delay), 3 (bad synchronization), 4 (lost) or 5 (overlap). If the code corresponds to a timeout, a loss or an overlapping, the PC will deal with the situation. To control the correct performance of the camera, the state of the camera is sent in both messages. In this way, the management workflow will release the camera control to improve performance in case of errors.

D. Final testing

The proposed software has been implemented in C#, and it has been tested during development. In addition to the testing carried out in the development stage, we defined a final test to check the whole software performance in a controlled scenario. The different tests were grouped in four main categories to check the communication, changes between different projects, wiring, and predefined protocols. All these tests have been

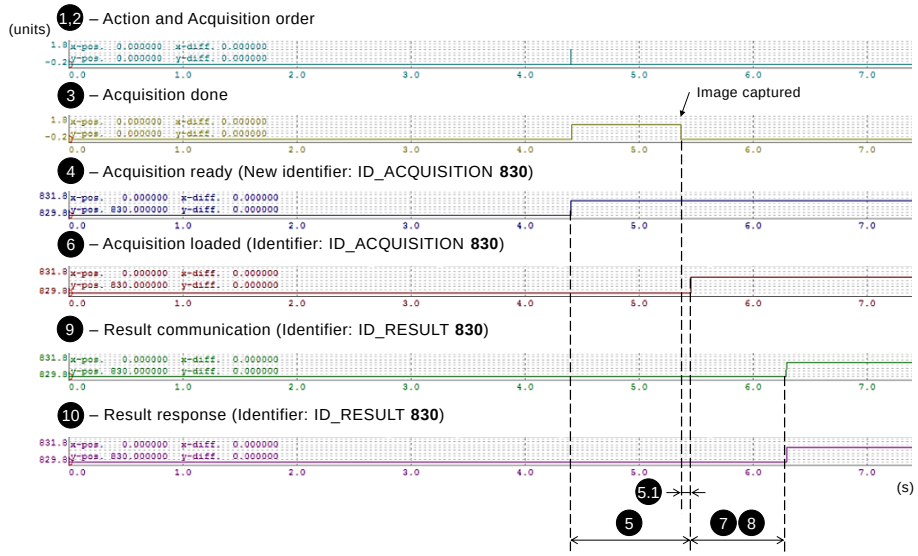
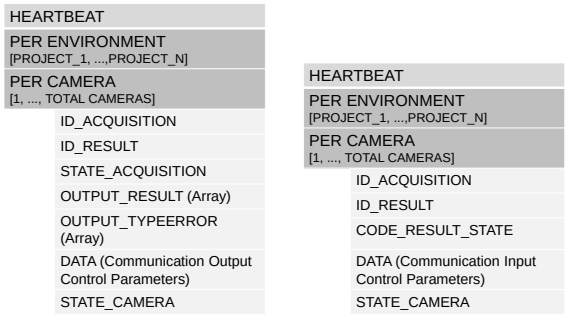


Fig. 5: Real-time scheduling together with the sequence of actions of the PLC, the PC, and the camera where numbers indicate the execution orders.



(a) (b)

Fig. 6: Messages from (a) PC to PLC, and from (b) PLC to PC.

registered in a checklist that is evaluated before installing the proposed software in a company.

1) Communication tests: PLC and PC communication is performed via messages as illustrated in Figures 4 and 6. These messages can be of different sizes depending on the context. To perform correctly, received and sent messages have to be the same size. Therefore, the first test checks that messages have the same size. Once this test is passed, the technical information, defined in the Communication Module of the Core level, is checked to control that communication is performed in the correct way. Afterwards, once communication is successfully established, possible interference, wire disconnections or other connectivity problems are checked via heartbeat. Finally, the information of the messages related to possible timeouts, delays, bad synchronization, losses or overlaps is analyzed to evaluate time performance by checking the viability and possible problems.

2) Changes between different projects: Changes between projects can be done via the PC Selection interface or via PLC. Regardless of the change's origin, the PC and the PLC must have the same information to properly control the context. Changes have to be consistent with the Working Mode of the Configuration Module. As described before, the PC has a register with the last active project of each context environment that will be communicated to the PLC. To check the correct performance, three situations have been considered. The first situation happens when the application starts (see Figure 7). The different situations that may arise, which depend on the PC and PLC active projects, have been checked to ensure correct performance. These situations are: (i) the PC has last projects and the PLC has no active projects, in which case the PC projects are applied; (ii) the PC has no last projects and the PLC has active projects, in which case the projects of the PLC are applied; (iii) the PC has no last projects and the PLC has no active projects, in which case no projects are applied; (iv) the PC has last projects and the PLC has the same ones, in which case no synchronization is required; and (v) the PC and the PLC have different projects, in which case the projects have to be synchronized following the defined master/slave policy. The second situation happens when the project received from the PLC/PC does not exist in the PC/PLC. In this case, the PLC/PC maintains its state and the PC/PLC will not receive any change. If this situation is given when the application starts and the PC is slave, no project will be loaded. If the PC is master or the project change is done from the application, the PC will wait for the PLC change and, if there is no change, this will be communicated via interface. The third situation happens when the project change occurs during application execution. In this case, acquisition loadings are stopped and only the processing workflows that started before the project change are ended. The application has to support changes at running time to perform efficiently.

3) Wiring tests: These tests have been designed to verify the application response to wiring problems. The possible situations have been simulated to check the application re-

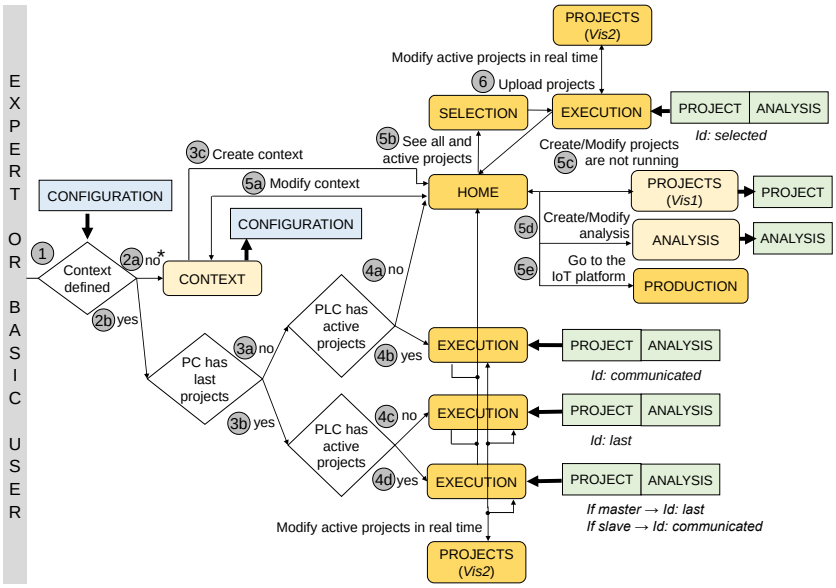


Fig. 7: Diagram of the proposed software steps. (*) The basic user can interact with the software when the context has already been defined by the expert, otherwise the application will be blocked.

sponse. First, the PLC and then the PC power has been cut and recovered. After power recovering, the application has checked the projects of both sides and has proceeded as described in the previous subsection. The same situation has occurred when the PC and PLC network cable has been disconnected. The next simulation has been focused on camera power disconnections checking the proper performance of the camera reactivation protocol. The last case has considered camera errors due to network disconnections. In this case, it is checked that the timeout is applied to reject the previous acquisitions that may have been ordered and could not be obtained. Although the timeout is applied, it should be noted that there is a set of images neither acquired nor processed.

E. Predefined protocol tests

The last group of tests verifies the proper performance of predefined protocols to control cameras, resources, procedures and possible errors during execution. For each protocol, all the steps that compose it are reviewed to ensure a proper performance.

III. Results

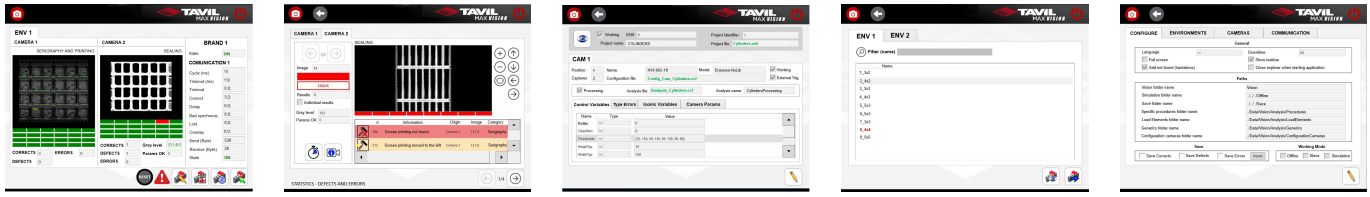
The proposed software has been installed in three different companies, two in the food sector and one in the paint industry. In addition, an in-house test has been performed (see Figure 8). In this section the main details of these installations are presented.

In all cases, time restrictions have been imposed by our industrial partners who wanted to reproduce the current manual performance using automatic processes. For this reason, their demand was to achieve the same production rates but with no manual interaction. Therefore, for each one of the use cases, the time restriction has been set to the time achieved with manual performance (see Table I, Max. Time row). As no similar systems were available to compare with, the effectiveness has been considered from the point of view of the computer vision success. It has been defined in terms of the number of evaluated products and the time required to evaluate them. For a better understanding of the use cases, the first one will be presented with all the details, while for the others the same information will be given in a summarized way.

A. Food industry

The first installation has been carried out in a company in the food industry to control the quality of product packaging, verify that they comply with manufacturing and regulatory specifications, and also to control the picking process. Four types of product (pizza dough, pizza, bacon and loin) have been considered. Each product will be presented as a different case, with the pizza dough having the most detailed description.

1) Pizza dough: For the quality control of pizza dough, which can be of different brands and with different packaging features, the machine vision system (MVS) has been installed inside a sealing machine that joins the upper film with the pizza dough package before the cutting process (see Figure 9). The system examines pizza dough packages to guarantee that manufacturing and regulatory specifications are met. Particularly, it checks the correct position of the screen printing (serigraphy test), the good visibility of the date and the lot (printing test), and the correct package closure and seal (sealing test). In this case, there are two environments, each one with one lineal color camera (CC), which performs the serigraphy and printing tests, and one lineal monochrome camera (MC), which performs the sealing test. There are as many scenarios as pizza brands. The details of the two environments are illustrated in Figure 10. As it can be seen, the main features of the context are the eight products that



(a) (b) (c) (d) (e)

Fig. 8: Interfaces of the proposed software corresponding to: (a) the Execution interface from the Food industry Bacon case; (b) the functionality from Execution interface to detect and explore errors and warnings in detail from the Food industry Pizza dought case; (c) the Projects (Vis2) interface from the In-house product; (d) the Selection interface from the Yogurt company; and (e) the Context interface.

TABLE I: Main details of the context and project description for the different companies where the software has been installed. Each column corresponds to a different company, except for the second, third, fourth and fifth columns, that correspond to different products of the same company. The acronyms are: Machine vision system (MVS), monochrome camera (MC), color camera (CC), field of view (FV), shape model (SM), not applicable (NA), and convolutional neural network (CNN).

CONTEXT DESCRIPTION							
Product	Dough	Pizza	Bacon	Loins	Yogurt Box	Paint	In-house
MVS place	Join machine	Conveyor	Join machine	Conveyor	Conveyor	Conveyor	Static
Action	Rejection	Rejection	Picking	Picking	Rejection	Picking	Picking
Environment	2	1	1	1	2	1	1
Camera type	Lineal CC Lineal MC	Lineal MC	Lineal CC Lineal MC	3D	CC	MC	3D
#Products FV	8	1	24	0...N	1	1	0...N
Scenarios	1...N	1...N	1...N	1	1...N	1	1
PLC to PC	NA	NA	NA	NA	NA	NA	Cylinder (A or B)
PC to PLC	NA	NA	NA	3D Position	NA	2D Position	3D Position
Max. Time	$8s$	$500ms$	$6s$	$5s$	$3s$	$2s$	NA
PROJECT DESCRIPTION							
CAMERA ACQUISITION MODE							
Parameters	Exposure Gain	Exposure Gain	Exposure Gain	Internal	Exposure	NA	Conf. 2D Conf. 3D
#Orders *#Captures* *#Lines*	2 (MC) 1 2 (MC)	1 1 3	2 (MC) 1 2 (MC)	1 2 NA	1 1 NA	1 1 NA	1 2 NA
State	NA	NA	NA	NA	NA	NA	NA
CAMERA EXECUTION MODE							
Preprocess	Yes (MC)	Yes	Yes (MC)	NA	NA	NA	NA
Analysis	2D SM	2D SM CNN	2D SM (CC) CNN (MC)	Regions	2D SM	Regions	3D SM
Output *#Results* *#Type errors*	8 3 per test + subcases	1 per step	24 3 per test + subcases	1 per step	1 2	1 1	1 per step

appear in the field of view (FV) and the two cameras. In order to support the different brands, the camera Acquisition Mode has to set exposure and gain parameters to fit the brand changes in film density, color and composition for each camera and according to each active project. There is a single capture per acquisition for the two cameras, but when it comes to the monochrome camera, two lines of different exposure are captured for each line of the acquired image. Focusing on the camera Execution Mode, two pre-processes are carried out for the monochrome camera: the first one builds a mosaic with the images acquired by the two consecutive acquisition orders, since the field of view does not cover the entire space to be analyzed; the second one breaks the mosaic image into two images with different exposure. For the color camera no pre-processes are required. The analyses use Halcon library [48], and are based on 2D shape models (SM) and regions entered as iconic input parameters that are selected according to the brand to be processed (see Figure 11). For each analysis eight results are obtained, which are considered together. The result is true if all camera's tests are correct, and false otherwise. There are eight values per analysis to indicate whether one type error occurred for a product in some of the tests (serigraphy, printing or sealing) or subcases, such as horizontal and vertical scrolling. In this case, a rejection occurs when any of the products that are within the field of view do not meet the specifications.

The main steps that compose the serigraphy and printing tests and the sealing test are represented in Figure 11. Details of the image processing techniques that are applied are presented for each step, as well as images to illustrate how they perform.

In this use case, no extra information is required neither from the PLC to the PC, nor from the PC to the PLC. Processing has to be done in 8 seconds, which corresponds to the available feed time of the sealing machine that joins the upper film. The time required by the PC to perform all the actions is presented in Figure 12.

The details of the defined project are summarized in the

Fig. 9: Two views of the machine vision system installed inside the sealing machine that joins the upper film with the pizza dough package.

second column of Table I.

2) Pizza: This case is presented in the third column of Table I. The machine vision system has been installed in the clean room just after the sealing machine that joins the upper film with the pizza package, and before other control processes. A conveyor belt transports pizzas and the system has to examine them to reject the ones that do not meet the defined quality requirements with a maximum process time of $500ms$ between pizzas. There is one environment with a lineal monochrome camera with a single pizza in the field of view. As in the previous case, as many scenarios as pizza brands are defined. The processing does not require extra information neither from the PLC to the PC, nor from the PC to the PLC. The main features of the project are also presented in Table I. Focusing on the camera Acquisition Mode, the parameters field requires exposure and gain to be defined to fit the changes in film density, color, and composition according to the active project. There is a single capture per acquisition, a single acquisition order, and three lines of different exposure are captured for each line of the acquired image. Focusing on the camera Execution Mode, there is a pre-processing to break the initial acquired image into three images of different exposure (see Figure 13(a)). The analysis is based on 2D shape models and regions entered as iconic input parameters that are defined according to the brands. The analysis uses a convolutional neural network (CNN), which is also given as an iconic input parameter. A single result is returned, which is true if the closure and seal validation is correct, and false otherwise. The analysis is composed of different tests that can return a specific type error in case of failure.

3) Bacon: This case is presented in the fourth column of Table I. To process bacon there is one environment with one lineal monochrome camera and one lineal color camera. The machine vision system has been installed inside a sealing machine that joins the upper film with the bacon package to control that manufacturing and regulatory specifications are met. The first camera is placed before the internal cutting process, while the second one is placed after it. As in the pizza dough case, three tests are defined: the serigraphy, the printing and the sealing tests. The first two tests are carried out with the first camera, and the third one with the second camera. Twenty-four separated products appear in the filed of view (see Figure 13(b)). If any of the twenty-four products does not meet the specifications, a robotic arm picks it up to reject it. To proceed, no extra information is required neither from the PLC to the PC, nor from the PC to the PLC. Since the position of the twenty-four products only varies in displacement, this can be determined by the movement of the conveyor belt and, hence, the robotic arm does not require extra information. As in the previous cases, different brands are produced and one scenario is required for each one. Processing has to be done in less than 6 seconds, which corresponds to the available feed time. The details of the defined project are described below. Focusing on the camera Acquisition Mode, as in the previous cases, the parameters field requires the exposure and gain for each camera according to the active project. There is a single capture per acquisition and, when it comes to the monochrome camera, two lines of different exposure are captured for each line of the acquired image. Focusing on the camera Execution Mode, a two-step pre-processing is required for the monochrome camera to first build a mosaic with two acquired images, and then break it into two images with different exposure. The field of view of the monochrome camera does not cover the entire space to be analyzed. In the case of the color camera, no pre-processes are required. The analysis of the first camera is based on 2D shape models and regions entered as iconic input parameters that vary according to the brand. The analysis of the second camera uses a convolutional neural network, which is also given as an input parameter.

Twenty-four results are obtained per analysis and product, being true only if all the camera's tests are correct, and false otherwise. There are twenty-four values per analysis and three possible type errors for each test, and also for some other considered sub-cases.

In Figure 8(a) the Execution interface of this case is presented. The screen shows the last processed acquisitions of the two cameras, labeled as camera 1 and camera 2, and also the information related to the number of correct and rejected products, including the number of errors accumulated. Focusing on the camera images, note that, at the top, the image title indicates the performed test (serigraphy and printing for image 1 from camera 1, and sealing for image 2 from camera 2). The iconic parameters are also represented in the images (in green and yellow for the first image, and in red for the second one). At the bottom of the image, twenty-four squares appear representing the twenty-four results of the processed products (green square for true result and red for false). On the right side of the screen, the information of the PC and PLC communication is shown, as well as other information related to the machine. Note that the output control parameters are also shown for Camera 2: the first one indicates the grey level, and the second one indicates the correctness of the input parameters, which are represented in green if they are correct and in red otherwise.

4) Loins: The last case of this food company is defined to process loins. As stated in the fifth column of Table I, there is one environment with one 3D camera and a conveyor belt.

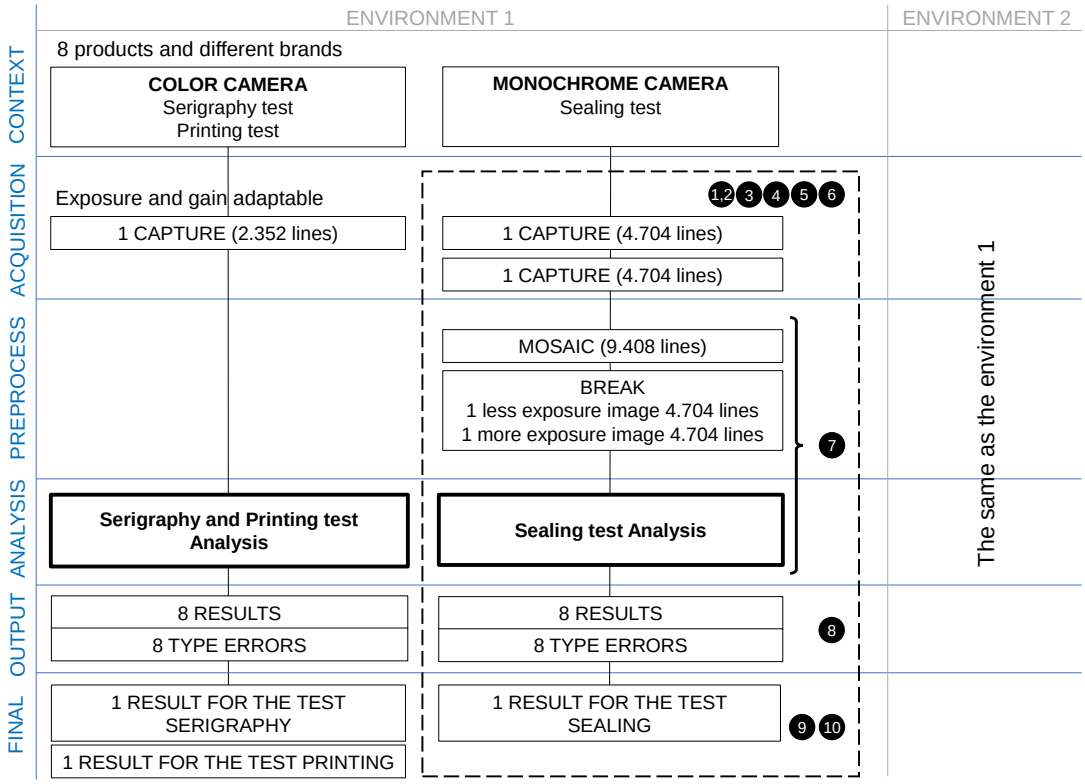
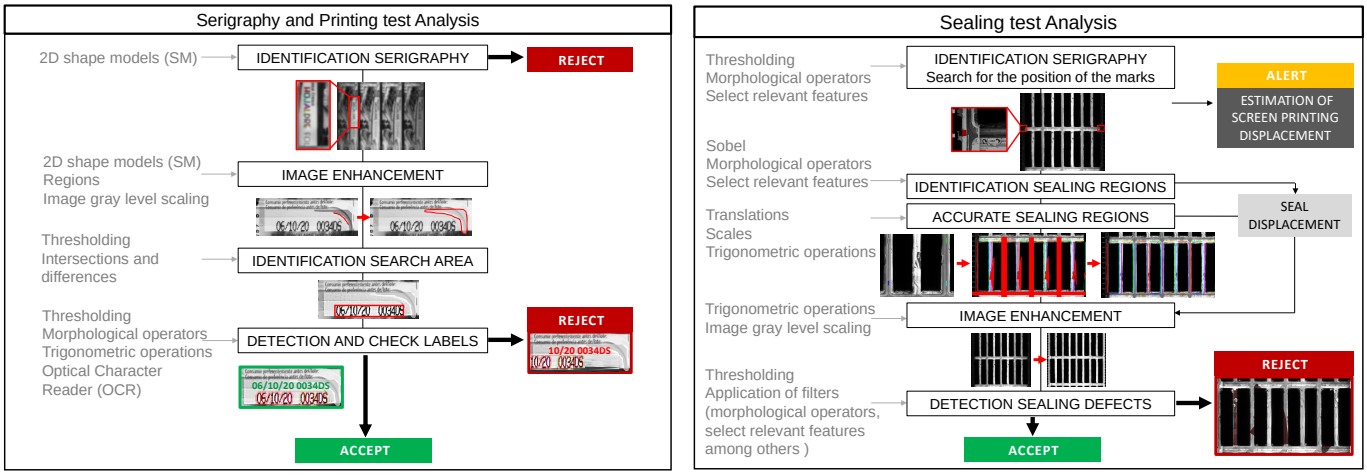


Fig. 10: Diagram of the different steps that define the solution for the Food industry Pizza dough case. The numbers indicate the execution orders represented in Figure 12, where the real-time scheduling together with the sequence of actions is illustrated.

Loin boxes are transported through the conveyor belt and a robotic arm empties them picking the loins up one by one. In the field of view there is the set of loins, which are inside the box and may decrease according to the picking process performed by the robotic arm. Only one scenario is required. The 3D position of the loin to be picked up is informed from the PC to the PLC. No extra information from the PLC to the PC is required. Processing has no time restrictions, although the time is considered acceptable if the box is emptied in less than 5 seconds. The details of the defined projects are described below. Focusing on the camera Acquisition Mode, the settings field does not require anything as the camera has its own internal management to acquire one 2D and one 3D image for each acquisition order. Focusing on the camera Execution Mode, no pre-processing is required. The analysis is based on regions to localize the loins and extract their 3D information to determine the best loin to pick up. The obtained result is true if there is a loin to pick up and its position is known, and false otherwise. There is one error type for each step of the analysis.

B. Yogurt company

The second installation has been done in a food company that produces yogurts. These are of different sizes and different diameters and are placed in boxes for distribution. The number of products per box differs according to the type of yogurt. In this company, the machine vision system has to reject boxes that do not contain the correct number of yogurts. The main details of this installation are presented in the sixth column of Table I. As indicated in the context description rows, the company has two environments, each one with a conveyor belt and a color camera. The machine vision system is placed on the conveyor belt to control the rejection process. In the field of view, a single box of yogurts appears (see Figure 13(d)). The context can support more than one scenario, each one defined to fit a specific yogurts box distribution. No extra information related to input or output control parameters is required neither from the PLC to the PC, nor from the PC to the PLC. Processing has to be done in less than 3 seconds, which corresponds to the minimum available time between boxes. This context description is registered in the Core level and Configuration Module of Figure 2. To determine how the specific scenario has to proceed, a project has to be defined. The main details of one project are given in the project description rows of Table I. The project defines the information to be stored in the Acquisition Mode and Execution Mode of the Camera Module, and the Execution Mode of the Environment Module. Focusing on the camera Acquisition Mode, the parameters field requires exposure to be defined to fit the yogurt colors and silkscreen features. There is a single capture per acquisition and a single acquisition order. Focusing on the camera Execution Mode, no pre-processing is required. Regarding the analysis, it is based on 2D shape models, which vary according to the yogurt features and are loaded according to the active project. The obtained results are true if the number of yogurts is the correct one and false



(a) (b)

Fig. 11: Main steps of the (a) serigraphy and printing tests and the (b) sealing test with details of the image processing techniques for the Food industry Pizza dough case.

otherwise. Two type errors are possible: a general error and a matching process error.

C. Paint factory

The third installation has been done in a paint factory. Paint is distributed in pots of different heights and with a handle. As expressed in the seventh column of Table I, the company has one environment with a conveyor belt and one monochrome camera. The machine vision system is placed on the conveyor belt to know the position of the paint pots in order for the robotic arm to pick them up. In the field of view, a single pot appears (see Figure 13(e)). The context only has one scenario since paint pots only differ in their heights, which cannot be detected by a 2D camera. To operate, the PC communicates the 2D position of the paint pot to the PLC and this one communicates it to the robotic arm to proceed. No extra information from the PLC to the PC is required. Processing has to be done in 2 seconds, which corresponds to the minimum available time between paint pots. Regarding the project description, the camera acquisition parameters are initially set and no changes are required. There is a single capture per acquisition and a single acquisition order. Focusing on the camera Execution Mode, no pre-processing is required. The analysis is based on paint pots region localization and position identification. No input iconic parameters are required. The obtained results are true if the pot position is known and false otherwise. There is also one possible type error.

D. In-house product

The last installation has been done in the company where the proposed software has been developed, which is specialized in factory mechanization. The machine vision system evaluates the position of cylinders to identify the highest one, which is the easiest to pick up by a robotic arm once its 3D position is known. As summarized in the last column of Table I, there is one environment with one 3D camera. The field of view covers some cylinders, which decrease during the picking process. There are two types of cylinders (type A and type B) that can appear at any moment with non-continuous flow. To avoid a continuous project change if two scenarios are considered, a single one is defined and the change in cylinder type (A or B) is controlled by the PLC, who informs the PC using the proper input control parameters. The PC will communicate the 3D position of the cylinder to be picked up to the PLC. Processing has no time restrictions. Regarding the project details, two captures (a 2D one and a 3D one) are acquired for each acquisition order after changing the camera parameters (see Figure 13(f)). There is a set of parameters to acquire 2D captures and another for 3D captures. Focusing on the camera Execution Mode, no pre-processing is required. The analysis is based on 3D shape models that localize the cylinders and extract their 3D information to determine the best one to be picked up. The obtained results are true if there is one cylinder to be picked up and its position is known, and false otherwise. There are some type errors that are specific to the analysis steps.

E. Limitations

The evaluated use cases have been useful to show the good performance of the proposed system. Unfortunately, no similar systems are available to compare in order to determine its effectiveness. In our case, the goal of the system was to replace manual procedures without increasing the processing time. This goal has been achieved, and in some cases the time has been reduced. For instance, the manual performance in the Food industry Pizza use case reaches a speed of $0.5m/s$, while the automatic approach reaches $1,0m/s$. Moreover, regarding the number of evaluated products, and as opposed to the manual procedures, the proposed approach ensures that all the products are always evaluated without human bias, since the whole process is carried out automatically.

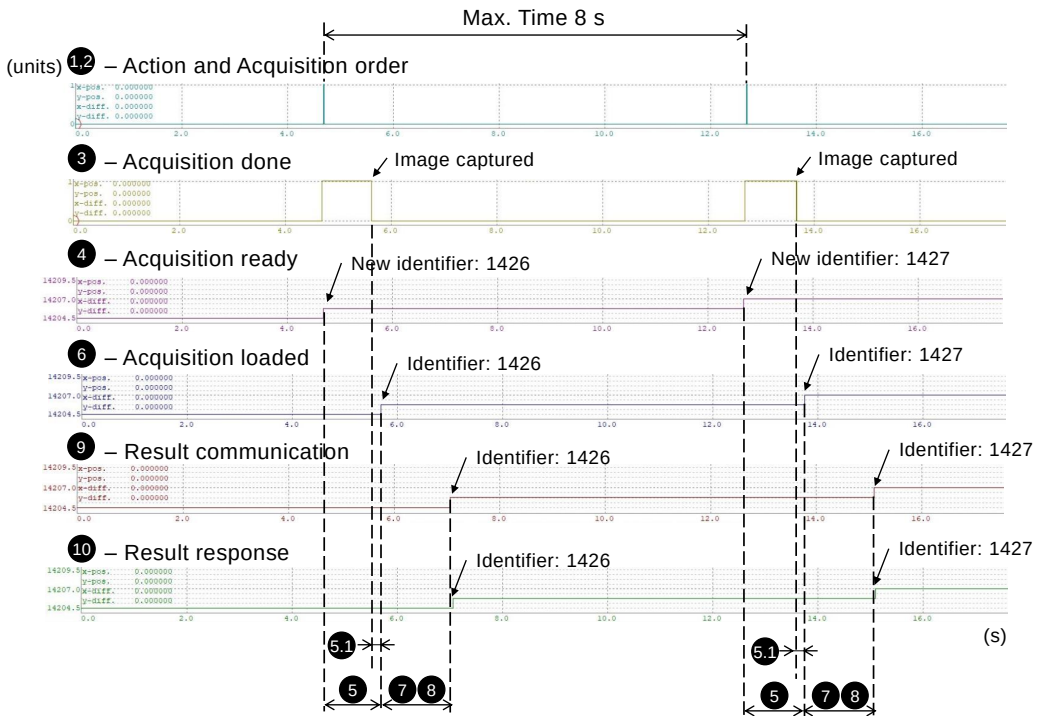


Fig. 12: Real-time scheduling together with the sequence of actions of the PLC, the PC, and the monochrome camera for the Food industry Pizza dough case. The numbers indicate the execution orders that have to be done in a period of eight seconds (Max. Time). $Result_{1426}$ (*Result communication* and *Result response*) has been obtained from both $Image_{1425}$, which has already been acquired, and $Image_{1426}$, whose acquisition process has been represented in this diagram (*Acquisition order*, *Acquisition done*, *Acquisition ready*, and *Acquisition loaded*). $Result_{1427}$ has been obtained from $Image_{1426}$ and $Image_{1427}$. The monochrome camera, in a pre-process, builds a mosaic with the images acquired by two consecutive acquisition orders.

Focusing on the time required to interact with the proposed software, as illustrated in Figure 2, two user profiles are supported: the expert user, who is responsible for the context configuration, and the basic user, who interacts at project level. For the former, a one-hour course is required to learn how to create the input resources required by the software to fit the requirements of the use case. For the latter, a fifteen minute training process carried out on site is enough to master the software. Although the learning curve may be thought as a limitation, the required time is considered to be acceptable.

IV. Conclusions and Future Work

Computer vision systems can benefit the industry at many different levels by reducing the degree of human interaction to achieve more efficient and effective procedures. However, the integration of these systems in a real context requires proper software able to control the different processes and also the different configurations that can appear according to the features of the processed products and the procedures to be applied.

In this paper, a software designed to manage the computer vision systems of a company has been proposed. To design it, the main features of a real company have been described using the following terms: context (all the mechanized space of a company controlled by computer vision), environment (part of a context with its own mechanical, computer vision and control components able to process a specific type of product), scenario (the environment processing a specific type of product with its own features and procedures) and project (specifications that determine how the environment components and processes have to proceed to support one scenario). These terms have been used to define the desired software requirements with special attention to the communication and interaction between components, the camera configurations, and the projects' variability. With the idea to fit as many companies as possible, a three-level architecture software has been proposed. The first level (core level) is the kernel of the software and has been designed to control the different environments, the vision system and all the components that define it, such as cameras and communication between components, among others. The second level (configuration level) has been designed to determine how the core level has to proceed according to the user needs, which vary with the context. The configuration is entered via user interfaces, which define the third level of the software.

Synchronization between all components and time restrictions have been analyzed and four classes that control the context, the environment, the camera and the results have been implemented and integrated into the software. As a result, a configurable and extensible software capable of fitting different scenarios by simply modifying a set of input parameters has been obtained. The software has been tested in a controlled scenario to check communication, changes between different projects, wiring, and predefined protocols. Once all the tests

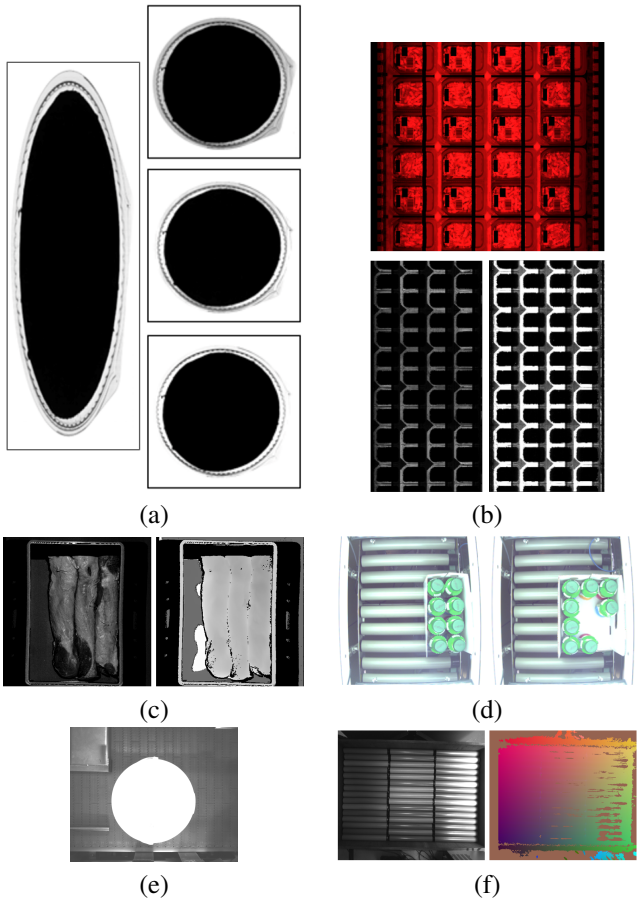


Fig. 13: Images of processed products corresponding to: (a) the pizzas from the Food industry, where the first image corresponds to the acquired image and the second group to the pre-processed images; (b) the bacon case from the Food industry, where the top image has been obtained with the color camera and the bottom ones with the monochrome camera (pre-processed); (c) the loins from the Food industry, where the images have been obtained from the same acquisition order and the camera returns a 2D capture and a 3D capture; (d) the Yogurt company; (e) the Paint company; (f) the In-house product, where the images have been obtained from the same acquisition order, one with a 2D configuration capture and the other with a 3D one.

have been passed, the software has been installed in four real companies with different scenarios and needs. For each one, the main details of the software installation have been described in order to show the adaptability and ease of use of the proposed software. These installations have also shown its good performance.

The current version of the proposed software covers the main requirements of a company. However, some modules need further development to fit more complex cases. The modular design of the software allows the integration of these new functionalities in a completely transparent way.

The details that have been given to present the proposed software can be understood as a guide for information system developers, since the main components and the main elements to be taken into account have been described.

Acknowledgments

This work has been carried out as part of the Industrial Doctoral program from the Catalan Government (2018 - 2021, Ref. 2018 DI 026). It has also been supported by grants from the Catalan Government (Nr. 2017-SGR-1101) and from the Spanish Government (Nr. MICIU PID 2019-106426 RB-C31).

References

[1] D. Sankowski and J. Nowakowski, *Computer Vision in Robotics and Industrial Applications*, ser. Series in Computer Vision: Volume 3. World Scientific, 2014.

[2] A. Stork, "Visual computing challenges of advanced manufacturing and industrie 4.0 [guest editors' introduction]," *IEEE Computer Graphics and Applications*, vol. 35, no. 2, pp. 21–25, 2015.

[3] J. Posada, C. Toro, I. Barandiaran, D. Oyarzun, D. Stricker, R. de Amicis, E. B. Pinto, P. Eisert, J. Döllner, and I. Vallarino, "Visual computing as a key enabling technology for industrie 4.0 and industrial internet," *IEEE Computer Graphics and Applications*, vol. 35, no. 2, pp. 26–40, 2015.

[4] L. Pérez, Í. Rodríguez, N. Rodríguez, R. Usamentiaga, and D. García, "Robot guidance using machine vision techniques in industrial environments: A comparative review," *Sensors*, vol. 16, p. 335, March 2016.

[5] C. Steger, M. Ulrich, and C. Wiedemann, *Machine Vision Algorithms and Applications*, 2nd ed. Wiley, 2018.

[6] P. Larrañaga, D. Atienza, J. Diaz-Rozo, A. Ogbechie, C. E. Puerto-Santana, and C. Bielza, *Industrial Applications of Machine Learning*, 1st ed. CRC Press, 2019.

[7] J. Lee, *Industrial AI, Applications with Sustainable Performance*, 1st ed. Springer Singapore, 2020.

[8] T. Brosnan and D.-W. Sun, "Improving quality inspection of food products by computer vision—a review," *Journal of Food Engineering*, vol. 61, pp. 3–16, 2004.

[9] D. Vernon, "The Space of Cognitive Vision," *Cognitive Vision Systems - Sampling the Spectrum of Approaches*, pp. 7–24, February 2006.

[10] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2011.

[11] T. B. Moeslund, *Introduction to Video and Image Processing, Building Real Systems and Applications*, 1st ed., ser. Undergraduate Topics in Computer Science. Springer-Verlag, 2012.

[12] A. Andreopoulos and J. K. Tsotsos, "50 Years of object recognition: Directions forward," *Computer Vision and Image Understanding*, vol. 117, pp. 827–891, 2013.

[13] L. Bodenhagen, A. R. Fugl, A. Jordt, M. Willatzen, K. A. Andersen, M. M. Olsen, R. Koch, H. G. Petersen, and N. Krüger, "An adaptable robot vision system performing manipulation actions with flexible objects," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 3, pp. 749–765, 2014.

[14] B. R. Mehta and Y. Jaganmohan Reddy, *Industrial Process Automation Systems*, 1st ed. Butterworth-Heinemann, 2014.

[15] L. Wang, M. Liu, and M. Q. . Meng, "Real-time multisensor data retrieval for cloud robotic systems," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 507–518, 2015.

[16] T. de Souza Alves, C. S. de Oliveira, C. Sanin, and E. Szczerbicki, "From Knowledge based Vision Systems to Cognitive Vision Systems: A Review," *Procedia Computer Science*, vol. 126, pp. 1855–1864, 2018.

[17] D. De Gregorio, R. Zanella, G. Palli, S. Pirozzi, and C. Melchiorri, "Integration of robotic vision and tactile sensing for wire-terminal insertion tasks," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 2, pp. 585–598, 2019.

[18] H. Fischer, M. Vulliez, P. Laguillaumie, P. Vulliez, and J. P. Gazeau, "Rtrobmultiaxiscontrol: A framework for real-time multiaxis and multirobot control," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 3, pp. 1205–1217, 2019.

[19] R. Syed, S. Suriadi, M. Adams, W. Bandara, S. J. J. Leemans, C. Ouyang, A. H. M. ter Hofstede, I. van de Weerd, M. T. Wynn, and H. A. Reijers, "Robotic Process Automation: Contemporary themes and challenges," *Computers in Industry*, vol. 115, p. 103162, 2020.

[20] S. Doltsinis, M. Krestenitis, and Z. Doulgeri, "A machine learning framework for real-time identification of successful snap-fit assemblies," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 1, pp. 513–523, 2020.

[21] L. Yang, I. Paranawithana, K. Youcef-Toumi, and U. Tan, "Confidence-based hybrid tracking to overcome visual tracking failures in calibration-less vision-guided micromanipulation," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 1, pp. 524–536, 2020.

[22] M. Lückenhaus, "Machine Vision in IIoT, How machine vision technologies help to overcome new challenges related to connected and automated production." available at https://www.qualitymag.com/articles/93296-machine-vision-in-iiot/, 2016, (Accessed: 10 December 2020).

[23] G. Aceto, V. Persico, and A. Pescapé, "A survey on information and communication technologies for industry 4.0: State-of-the-art, taxonomies, perspectives, and challenges," *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. 3467–3501, 2019.

[24] P. Martinez, M. Al-Hussein, and R. Ahmad, "A scientometric analysis and critical review of computer vision applications for construction," *Automation in Construction*, vol. 107, p. 102947, 2019.

[25] V. Kakani, V. H. Nguyen, B. P. Kumar, H. Kim, and V. R. Pasupuleti, "A critical review on computer vision and artificial intelligence in food industry," *Journal of Agriculture and Food Research*, vol. 2, p. 100033, 2020.

[26] M. Al-Sarayreh, M. M. Reis, W. Q. Yan, and R. Klette, "Potential of deep learning and snapshot hyperspectral imaging for classification of species in meat," *Food Control*, vol. 117, p. 107332, 2020.

[27] K. Yanai and Y. Kawano, "Food image recognition using deep convolutional network with pre-training and fine-tuning," *IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pp. 1–6, 2015.

[28] J. Kvam, L. E. Gangsei, J. Kongsro, and A. H. Schistad Solberg, "The use of deep learning to automate the segmentation of the skeleton from CT volumes of pigs," *Translational Animal Science*, vol. 2, no. 3, pp. 324–335, May 2018.

[29] A. Gila, M. Bejaoui, G. Beltrán, and A. Jiménez, "Rapid method based on computer vision to determine the moisture and insoluble impurities content in virgin olive oils," *Food Control*, vol. 113, p. 107210, 2020.

[30] M. X. Bastidas-Rodriguez, L. Polania, A. Gruson, and F. Prieto-Ortiz, "Deep learning for fractographic classification in metallic materials," *Engineering Failure Analysis*, vol. 113, p. 104532, 2020.

[31] R. Pereira, V. Filho, L. Moura, N. A. Kumar, A. Alexandria, and V. Albuquerque, "Automatic quantification of spheroidal graphite nodules using computer vision techniques," *The Journal of Supercomputing*, vol. 76, 2020.

[32] R. Benotsmane, L. Dudás, and G. Kovács, "Survey on new trends of robotic tools in the automotive industry," in *Vehicle and Automotive Engineering 3*, K. Jármai and K. Voith, Eds. Singapore: Springer Singapore, 2021, pp. 443–457.

[33] S. Unnikrishnan, J. Donovan, R. Macpherson, and D. Tormey, "Machine learning for automated quality evaluation in pharmaceutical manufacturing of emulsions," *Journal of Pharmaceutical Innovation*, vol. 15, pp. 1–12, 2019.

[34] B. Gallagher, M. Rever, D. Loveland, T. N. Mundhenk, B. Beauchamp, E. Robertson, G. G. Jaman, A. M. Hiszpanski, and T. Y.-J. Han, "Predicting compressive strength of consolidated molecular solids using computer vision and deep learning," *Materials Design*, vol. 190, p. 108541, 2020.

[35] S. Cai, Z. Ma, M. J. Skibniewski, and S. Bao, "Construction automation and robotics for high-rise buildings over the past decades: A comprehensive review," *Advanced Engineering Informatics*, vol. 42, p. 100989, 2019.

[36] C.-J. Liang, S.-C. Kang, and M.-H. Lee, "RAS: a robotic assembly system for steel structure erection and assembly," *International Journal of Intelligent Robotics and Applications*, vol. 1, pp. 459–476, 2017.

[37] Y.-S. Lee, S.-H. Kim, M.-S. Gil, S.-H. Lee, M.-S. Kang, S.-H. Jang, B.-H. Yu, B.-G. Ryu, D. Hong, and C.-S. Han, "The study on the integrated control system for curtain wall building façade cleaning robot," *Automation in Construction*, vol. 94, pp. 39–46, 2018.

[38] B. Wei, K. Hao, L. Gao, and X.-S. Tang, "Detecting textile micro-defects: A novel and efficient method based on visual gain mechanism," *Information Sciences*, vol. 541, pp. 60–74, 2020.

[39] C. Gonzalez-Val, A. Pallas, V. Panadeiro, and A. Rodriguez, "A convolutional approach to quality monitoring for laser manufacturing," *Journal of Intelligent Manufacturing*, vol. 31, pp. 789–795, 2020.

[40] Y. Li and Y. Zhang, "Application research of computer vision technology in automation," in *2020 International Conference on Computer Information and Big Data Applications (CIBDA)*, 2020, pp. 374–377.

[41] E.R. Davies, *Computer Vision: Principles, Algorithms, Applications, Learning*, 5th ed. Academic Press, 2018.

[42] Y. Zhang, Z. Jia, and Y. Dai, "Real-time performance analysis of industrial serial production systems with flexible manufacturing," in *2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS)*, 2018, pp. 360–365.

[43] E. R. Dougherty and P. A. Laplante, *Introduction to Real-Time Imaging*. SPIE Press/IEEE Press, 1995.

[44] H. Gomaa, "Advances in software design methods for concurrent, real-time and distributed applications," in *2008 The Third International Conference on Software Engineering Advances*, 2008, pp. 451–456.

[45] Q.-Y. Gu and I. Ishii, "Review of some advances and applications in real-time high-speed vision: Our views and experiences," *International Journal of Automation and Computing*, vol. 13, pp. 305–318, 2016.

[46] Y. Watanabe, O. Hiromasa, and M. Ishikawa, "Architectures and applications of high-speed vision," *Optical Review*, vol. 21, pp. 875–882, 2014.

[47] H. Gomaa, *Software Design Methods for Concurrent and Real-Time Systems*, 1st ed. USA: Addison-Wesley Longman Publishing Co., Inc., 1993.

[48] MVTec, "MVTec Software GmbH HALCON – The power of machine vision," available at https://www.mvtec.com/products/halcon/, (Accessed: August 2020).

[49] OpenCV, "Open Source Computer Vision Library," available at https://opencv.org/, (Accessed: December 2020).

[50] Cognex, "VisionPro Software," available at https://www.cognex.com/products/machine-vision/vision-software/visionpro-software/, (Accessed: December 2020).

[51] M. G. Bedia, M. Aguilera, L. F. Castillo, L. Uribe, M. J. Manrique, and G. Isaza", "Case-based reasoning and real-time systems: Exploiting successfully poorer solutions," in *2011 6th Colombian Computing Congress, CCC 2011*, May 2011, pp. 1–4.

[52] AIA Global Vision Systems Trade Association, "GigE Vision," available at https://www.visiononline.org/vision-standards-details.cfm?type=5/, (Accessed: December 2020).

[53] AIA Global Vision Systems Trade Association, "USB3 Vision," available at https://www.visiononline.org/vision-standards-details.cfm?id=167type=11, (Accessed: December 2020).

[54] EMVA European Machine Vision Association, "GenICam," available at https://www.emva.org/standards-technology/genicam/, (Accessed: December 2020).

[55] B&R Industrial Automation GmbH, "Communication and fieldbus systems," available at https://www.br-automation.com/en/products/software/additional-information/communication-and-fieldbus-systems/, (Accessed: December 2020).

Núria Banús BSc in Computer Science from the Universitat de Girona (UdG) in 2016 and MSc in Computer Science from UdG in 2018. She is currently a predoctoral researcher in an Industrial Doctoral program with the Graphics and Imaging Laboratory, assigned to the Computer Science and Applied Mathematics Department at UdG, and with R&D department of the TAVIL Ind. S.A.U. company within the computer vision team. She carries out R&D on image processing techniques applied to industrial processes.

Imma Boada Professor at Computer Science and Applied Mathematics Department at the Universitat de Girona. MSc in Computer Science from the Universitat Autònoma de Barcelona (1992) and PhD in Computer Science from the Universitat Politècnica de Catalunya (2001). She is a researcher of the Graphics and Imaging Laboratory where she carries out R&D on visualization, image processing, serious games and e-learning. She has special interest on medical applications and the automation of industrial processes.

Pau Xiberta BSc in Computer Science from the Universitat de Girona (UdG) in 2013, MSc in Computer Science from the Universitat Oberta de Catalunya in 2015, and PhD degree in Technology from UdG in 2018. He is currently a postdoctoral researcher with the Graphics and Imaging Laboratory, and assigned to the Computer Science and Applied Mathematics Department at UdG. His research interests include image processing techniques applied to industrial processes.

Pol Toldrà BSc in Industrial Electronics from the Universitat de Girona and MSc in Automatics and Industrial Electronics from the Universitat Rovira i Virgili. He has more than 20 years of experience in automation of industrial processes and he is the head of the R&D department from TAVIL IND S.A.U.