AGILEHAND

D4.1 – AGILEHAND Smart Sensing SUITE v1

WP4 – BUILD: AGILEHAND Smart Sensing SUITE

AGILEHAND has received the funding from the European Union's Horizon Europe research and

AGILEHAND

Document Information

GRANT AGREEMENT NUMBER	101092043	ACRONYM		AGILEHAND
FULL TITLE	Smart grading, handling, and packaging solutions for soft and deformable products in agile and reconfigurable lines			
START DATE	01-01-2023	DURATION		36 months
PROJECT URL	https://agilehand.eu/			
DELIVERABLE	D4.1 – AGILEHAND Smart Sensing SUITE			
WORK PACKAGE	WP4 – BUILD: AGILEHAND Smart Sensing SUITE			
DATE OF DELIVERY	CONTRACTUAL	June 2024	ACTUAL	-
NATURE	Other	DISSEMINATION LEVEL		Public
LEAD BENEFICIARY	FBK			
RESPONSIBLE AUTHOR	Dr. Mohamed Lamine Mekhalfi			
CONTRIBUTIONS FROM	FBK, UNI, UPM			
TARGET AUDIENCE	1) AGILEHAND Project partners; 2) industrial community; 3) other H2020 funded projects; 4) scientific community			
DELIVERABLE CONTEXT/ DEPENDENCIES	This document is the first iteration. Its relationship to other documents is as follows: D4.2 Title: AGILEHAND Smart Sensing SUITE v2			
EXTERNAL ANNEXES/ SUPPORTING DOCUMENTS	None			
READING NOTES	None			
ABSTRACT	The scope of D4.1 is to showcase the progress made so far within WP4 which regards four pilots. In this respect, the proposed solutions' smart sensing, data acquisition, quality grading, and self-calibrating aspects are reported and discussed. Further, we share functional and implementation viewpoints of the different functions envisioned in WP4.			

Document History

VERSION	ISSUE DATE	STAGE	DESCRIPTION	CONTRIBUTOR
0.1	24/04/2024	ToC and working version	Created ToC and document structure	FBK
0.2	03/05/2024	Working version	1st draft of the document (T4.2 and T4.3)	FBK, UNINOVA
0.3	04/06/2024	Working version	2nd draft of the document (T4.1)	FBK, UNINOVA
0.4	11/06/2024	Working version	3rd draft of the document (including a section about functional and implementation viewpoints)	FBK, UNINOVA, UPM
1.0	20/06/2024	Final draft version	Final version for internal review	FBK, UNINOVA
1.1	26/06/2024	Reviewed version	The final version of the document, internally reviewed	UPV, TAU
1.2	28/6/2024	1st after reviewing the version	Internal reviews addressed	FBK
1.3	30/06/2024	Final version	Internal reviews addressed	UNINOVA

Disclaimer

Copyright message

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS/ACRONYMS

AI	Artificial Intelligence
API	Application Programming Interface
CR^{Hand}	Collaborative Robot Handling
DS^{Sense}	Data Sets Sensing
DT^{Agile}	Data-Driven Digital Twin
ETA	Estimated time of arrival
GQ^{sense}	Grade the Quality Sensing
IoT	Internet of Things
ICT	Information and Communication Technologies
KER	Key Exploitable Result
KPI	Key Performance Indicator
MES	Manufacturing Execution System
ML	Machine Learning
OER	Other Exploitable Result
OPC/UA	Open Platform Communications/Unified Architecture
PE^{Agile}	Production Execution Optimization Toolkit
PLC	Programmable Logic Controller
PR^{Agile}	Production Reconfiguration
PT^{Agile}	Product Oriented Traceability
RR^{Hand}	Robot Robot Handling
SC^{sense}	Self-Calibrating Sensing
SMED	Single-Minute Exchange of Dies
ST^{hand}	Self-Adaptable Transportation System
UI	User Interface
WP	Work Package

Executive summary

This deliverable describes the activities carried out within WP4 under AGILEHAND pilots, namely **Multiscan**, **Sant'Orsola**, **Produmar** and **Marelec**. In particular, self-calibration, quality grading, dataset acquisition, and smart sensing solutions are described in the following sections.

Document structure

Section 1 describes the work on the "**AGILEHAND** Self-Calibrating" solution, developed in **Task 4.1**, describing the solution overview and the implementation status (current and future work). This section describes how this solution will acquire images to be used in product classification.

Section 2 describes the work on the "**AGILEHAND** Grade the Quality" solution, developed in **Task 4.2**, describing the solution overview and the implementation status (current and future work). This section describes how the solution uses machine learning to classify products to support the handling of soft and deformable products.

Section 3 describes the work carried out on the "**AGILEHAND** Data-Sets" solution, developed in **Task 4.3**, describing the solution overview and the implementation status (current and future work). This section describes how the images were classified to be used to train the automatic algorithm to detect the products.

Section 4 provides a technical description of the "**AGILEHAND** Smart Sensing" suite, describing it from a functional and implementation point of view.

Section 5 includes the conclusions of the document and the work carried out in the three tasks of **WP4**.

1. AGILEHAND Self-Calibrating

1.1. Overview

AGILEHAND is developing an intelligent product grading approach to identify different products very quickly and with little human effort. The grading is achieved through a set of sensors that help monitor various quality parameters across different pilot products as well as machine learning techniques, where the types of sensors and their positioning will be activated and deactivated according to the current product and task. In **Task 4.1**, the necessary sensors were identified and are described in the following sections.

1.2. Implementation Status

The sensorization requirements for each of the pilots and their respective solutions were surveyed twice (in January and May 2024). The survey was created to help identify pilot requirements and solutions needed to acquire data for this task. Since it is necessary to identify what type of data is needed for pilots and solutions, to choose the appropriate sensors. It was also asked where the solutions will be hosted, and whether they will be based on Cloud or Edge. However, there are still undefined requirements for some of the solutions. Thus, the technical requirements (hardware needs, sensors, software, operational conditions, interconnections, etc..) will continue to be iterated while the solutions are being developed.

1.2.1. Current Implementation

This section describes how the implementation of self-calibration in the pilots was defined and how it's being carried out for the different factories.

1.2.1.1. Multiscan

In the **Multiscan** pilot, the S90 model will be used (Figure 1). This system already has the Sony IMX429[1] camera incorporated, which is used to detect defects in food products (vegetables and fruits) and perform the grading task. However, in **AGILEHAND** we seek to tune the grading parameters of the machine according to the input batch of fruits to be graded. For instance, the aim is to infer the distribution (e.g., histogram) of the fruits of the input batch based on several criteria such as shape and size. To this end, another sensing solution will be put in place before the fruit batch is unloaded into the grading machine.

[1] https://www.sony-semicon.com/en/products/is/industry/global-shutter.html

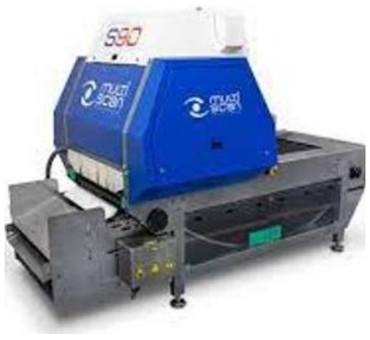
Figure 1: Multiscan S90.

1.2.1.2. Sant'Orsola

At **Sant'Orsola**, tests are being carried out with different types of cameras to identify which is the most suitable one for acquiring quality images for classification purposes. So far, tests have been carried out with the Intel RealSense™ D415[2], Intel RealSense™ D456[3] and a Stereo Vision (FLIR Blackfly BFLY-PGE-13E4C-CS[4]), but the desired results have not been obtained mainly due to the small size of the fruits. The next tests envision a LIDAR (along with Intel RealSense™ D456) sensor to enable high-quality depth image acquisition.

1.2.1.3. Produmar

The camera setup in Produmar uses the Intel RealSense™ D456 for the image data acquisition as illustrated in the next sections, as well as for fish steak cutting and classification. Furthermore, it was also used for identifying the positions of fish steaks and fillets, for robotic arm manipulation to feed the packaging machine.

Additionally, for the traceability solution (**Task 6.1**), a set of temperature-reading RFID tags is being sourced to support the tracking of the batches of fish along the cold chain in the production line. The specification of the tags needs to support working conditions of -40ºC and high humidity, which makes the use of passive tags ideal for the task. An initial set of tag suppliers have been identified and are being contacted to determine the most suitable option.

Figure 2 illustrates the floor plan of **Produmar's** production line and the respective placement of the different types of sensors.

[2] https://store.intelrealsense.com/buy-intel-realsense-depth-camera-d415.html
[3] https://www.intelrealsense.com/depth-camera-d456/
[4] https://www.flir.eu/products/blackfly-gige/?vertical=machine+vision&segment=iis

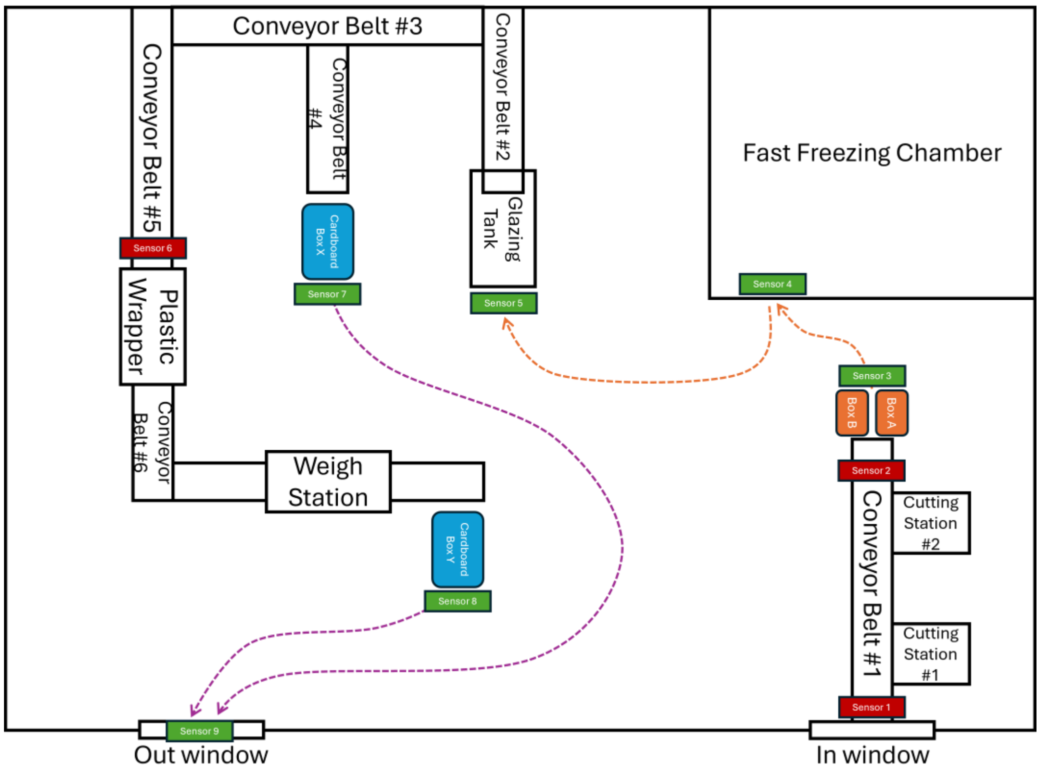

Figure 2: Produmar's floor plan and sensor placement.

1.2.1.4. Marelec

In Marelec's case, Intel RealSense^{TM} D456 cameras will be exploited for the image acquisition task to enable machine learning-based grading.

1.2.2. Future Developments

In the case of **Multiscan**, there are no further developments to be carried out in this task, as the machines are already equipped with cameras and **Marelec**, Intel RealSense^{TM} D456 will be explored to acquire RGB and depth images. Regarding **Sant'Orsola**, Intel RealSense^{TM} D456 will be exploited to acquire RGB images, and a LIDAR sensor will be used to acquire depth images. Further, **Produmar's** role will be to place the cameras with the BHealth Box (described in the deliverable D2.1) to send the datasets to the Cloud, to be used by **AGILEHAND** solutions.

2. AGILEHAND Grade the Quality

2.1. Overview

AGILEHAND aims at leveraging advanced artificial intelligence (AI) and machine learning technologies for grading, handling and packaging autonomously soft and deformable products, providing a strategic instrument to improve the flexibility, agility and reconfigurability of production and logistic systems to the European manufacturing companies.

In the context of the use cases being addressed in AGILEHAND, the handling and packaging endeavour involves Machine Vision (MV) solutions in coordination with a robotic manipulation system (addressed within **WP5** of AGILEHAND). In particular, the MV system will consist of 2D or 3D cameras that enable the system to see the target product, whilst the robotic system receives instructions from the vision system regarding the position and the grade of the target product and proceeds thereafter with the product handling and packaging operations. To highlight the harmony between the vision and the robotic solutions, we depict in Figure 3 an abstract representation of a fruit quality approval system. For instance, the MV system (on the right-hand side of the figure) provides information to the robotic manipulator to pick up and place the product under inspection for logistic processing. This information might regard the grade of the product (i.e., according to criteria such as health status, size, and ripeness) as well as location coordinates to enable its handling.

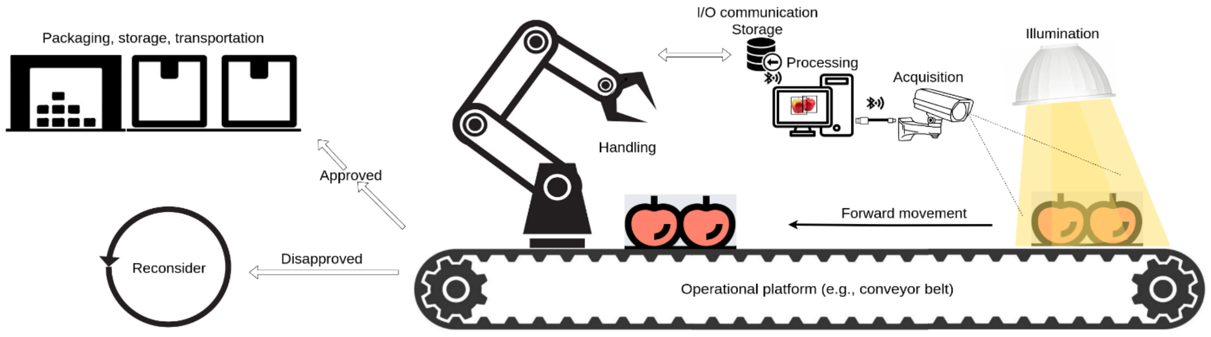

Figure 3: Machine vision in collaboration with a robotic system.

Yet, it is evident that the engagement of the MV and robotic solutions is synchronised to establish a smooth cooperation. This implies that the precision information provided by the MV system (e.g., product coordinates, status, grade) must be reliable to ensure correct robotic handling and mitigate errors. This calls for the adoption of cutting-edge object detection and grading techniques. The relevant literature suggests many contributions in this respect [1]-[6]. Overall, existing works can be viewed from two perspectives, namely (i) traditional image processing techniques, which typically implement thresholding and pixel-level analysis, and (ii) data-driven techniques that train models on a specific task by feeding images along with annotation labels. While traditional methods are straightforward, they often suffer from generalisation bottlenecks when slight changes in illumination are encountered, owed mainly to the thresholding settings being determined according to a specific illumination scenario. Machine learning methodologies, however, do not require manual thresholding or handcrafted analysis, which renders them less prone to generalisation issues when slight illumination changes occur. On the other hand, machine learning schemes require data (e.g., images and labels), which can be prohibitive and time-consuming.

Deep learning (DL), a subset of machine learning, consists of neural networks that encompass many layers (thus the word 'deep') that are connected via weights. Typically, a DL model is trained offline on a training dataset to achieve a certain task (e.g., detection, segmentation), and then the trained model is deployed online to carry out that specific task on unseen data. Owing to their nature of being 'deep', they normally require plenty of data to converge during training [7], [8]. However, thanks to transfer learning, a DL model that was pre-trained on a large dataset on a given task can be finetuned on another dataset of interest that does not possess large volumes of data [9] even if the task is different (e.g., a DL model that was pre-trained for the task of object classification on a large dataset such as ImageNet [10] can be tailored to address the task of object detection on a different dataset [11]).

Task 4.2 encompasses the work related to the grading and positioning of products with the support of machine vision and, in addition, supporting the robotic manipulation task that is carried out in WP5.

2.2. Implementation Status

This section demonstrates the current implementations and the next developments for each pilot and discusses the results and challenges where necessary.

It is important to note that this section regards the grading task, which is addressed by employing data-driven techniques. Therefore, to comprehend some concepts mentioned here, it is necessary to read also the next section that details the dataset acquisition part. Technically, **Task 4.3** comes first and is succeeded by **Task 4.2**.

2.2.1. Current Implementation

2.2.1.1. Multiscan

Multiscan decided to explore the problem of orange grading. They already have grading systems set up in place and they want to further improve their precision. As an initial feasibility analysis, we conducted an experimental study to assess the capability of recent machine learning paradigms in grading oranges. In particular, a grading pipeline based on deep learning has been developed to grade each orange fruit into one of three classes, namely Good, Bad, and Undefined, and assess the performance of the solution concerning each class. In particular, two deep learning models have been adopted. The first one is based on ResNet-18 architecture, which has proven useful for image classification [12]. However, the grading solution may eventually be deployed in real-time, which may require a deep model with fewer parameters. Thus, a second classification model based on SqueezeNet [13] has also been employed. The images are resized to the dimension of 2500x300 before being fed to the models.

The performance measure is expressed in terms of classification Accuracy per class, which indicates the sum of correctly classified samples over the total number of available samples. The average classification Accuracy across the three classes is also reported, as well as the Overall classification accuracy, which represents a classification Accuracy regarding the samples of all the classes put together.

First, Table 1 reports the scores when all the orange views are considered. It can be observed that the class Bad reports the highest score, followed by the Good class, and then the Undefined class. This is due to the imbalance in the number of samples per class, as summarised in the next sections (Table 4 of Section 3). In particular, the class Undefined contains very few samples, and this reflects a low classification rate. On the other hand, the overall classification is plausible, owed mainly to the high number of samples in the Bad class that reports a high accuracy.

It should also be noted that ResNet-18 performs better than SqueezeNet when classifying the Bad class, while the opposite is true for the Good class. This is perhaps because ResNet-18 is a larger model that performs well when abundant images are available. In terms of average classification accuracy, SqueezeNet performs better than ResNet-18 (+2.7%). Both models perform almost on par in terms of overall accuracy.

Table 1: Multiview classification scores (%).

	Good	**Bad**	**Undefined**	**Avg.**	**Overall**
ResNet-18	57.60	87.50	21.40	55.50	73.30
SqueezeNet	72.70	80.70	21.40	58.30	72.60

Second, to evidence the choice of Multiview orange classification, a further experiment was conducted by retaining only one view of each orange while discarding the remaining ones. The scores are given in Table 2. Although the classification score of the Bad class increases for both models, as this class features a higher number of samples, the scores of the other two classes have dropped significantly. This highlights the advantage of multiview image analysis for orange grading.

Table 2: Single view classification scores (%).

	Good	**Bad**	**Undefined**	**Avg.**	**Overall**
ResNet-18	39.40	88.60	7.10	45.10	68.10
SqueezeNet	42.40	94.30	0.00	45.60	71.90

To be noted that the scores summarised in Table 1 and Table 2 were obtained with models that were pre-trained on the ImageNet dataset. This is a common practice in deep learning image analysis to transfer the knowledge learned by a model on a certain dataset (typically a large one like ImageNet) to another dataset (or task). This is referred to as transfer learning and has been shown to incur improvements. Therefore, in a third experiment, both models were trained from scratch (without pretraining on ImageNet), resulting in the values reported in Table 3. Interestingly, the scores of the class Bad have improved drastically because of the high number of samples in this class. Regarding the Good class, ResNet-18 maintains its performance while SqueezeNet collapses. The classification of the Undefined samples does not seem possible in both models. Therefore, for datasets with a few samples, pretraining is essential.

As per the lower classification rates of the Good and the Undefined classes concerning the Bad class, and apart from the dataset imbalance, it is observed that the annotation of the dataset plays a fundamental role. For instance, two qualitative examples are depicted in Figure 4, where the first row shows instances of an orange that was annotated as Good, and the second row illustrates instances of an orange that was annotated as Undefined. Visibly the Undefined orange looks smoother and more uniform than the Good one.

Table 3: Multiview classification scores (%) with scratch-trained models.

	Good	Bad	Undefined	Avg.	Overall
ResNet-18	57.60	90.90	0.00	49.50	73.30
SqueezeNet	0.00	100.0	0.00	33.30	65.20

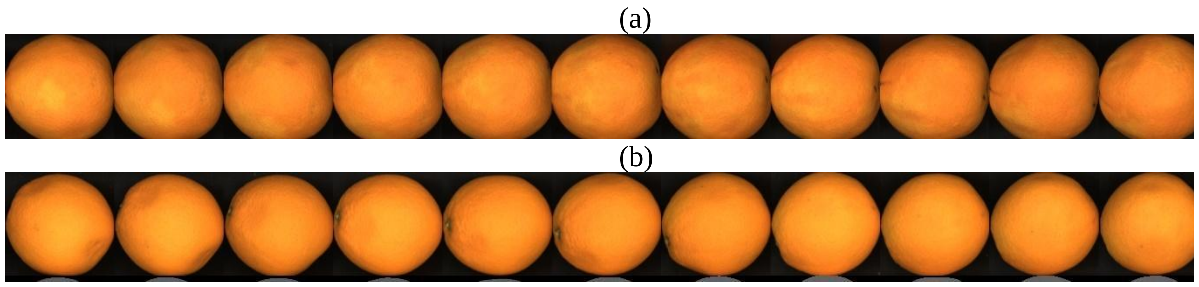

Figure 4: Annotation examples. (a) Good sample, and (b) Undefined sample.

2.2.1.2. Sant'Orsola

The grading task regarding **Sant'Orsola's** use case entails five raspberry grades, which are based on the maturity of the fruits. Precisely, punnets containing fruits of different grades pass along the conveyor belt, and the goal is to (i) identify the segment of every single fruit inside the punnet, (ii) grade it into one of five classes, and (iii) determine its precise coordinates to enable the robotic solution of AGILEHAND to handle it (e.g., pick the inadequate fruits out of the punnet and place it in a different point). The five grades of raspberries can be distinguished based on their colour. For instance, for the sake of clarification, we depict in Figure 5**,** five punnets, each one containing only fruits of a certain grade.

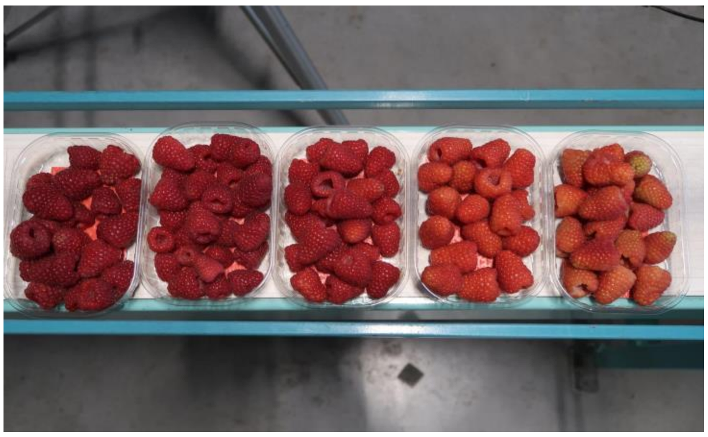

Figure 5: Examples from the five Raspberry grading classes.

In this respect, for the robotic arm to manipulate single fruits, the location and the depth of each single fruit (i.e., distance from the robotic picking arm to the fruits) is required. Moreover, instance segmentation of single fruits is necessary to enable the robotic system to address the target fruit without compromising the surrounding ones, which is a delicate process.

As detailed in the next section **(Task 4.3)**, the use of RealSense™ sensors to acquire the RGB and depth components simultaneously was explored. The RGB image will be adopted for the instance segmentation process, whilst the depth map will serve for the determination of the coordinates of each fruit.

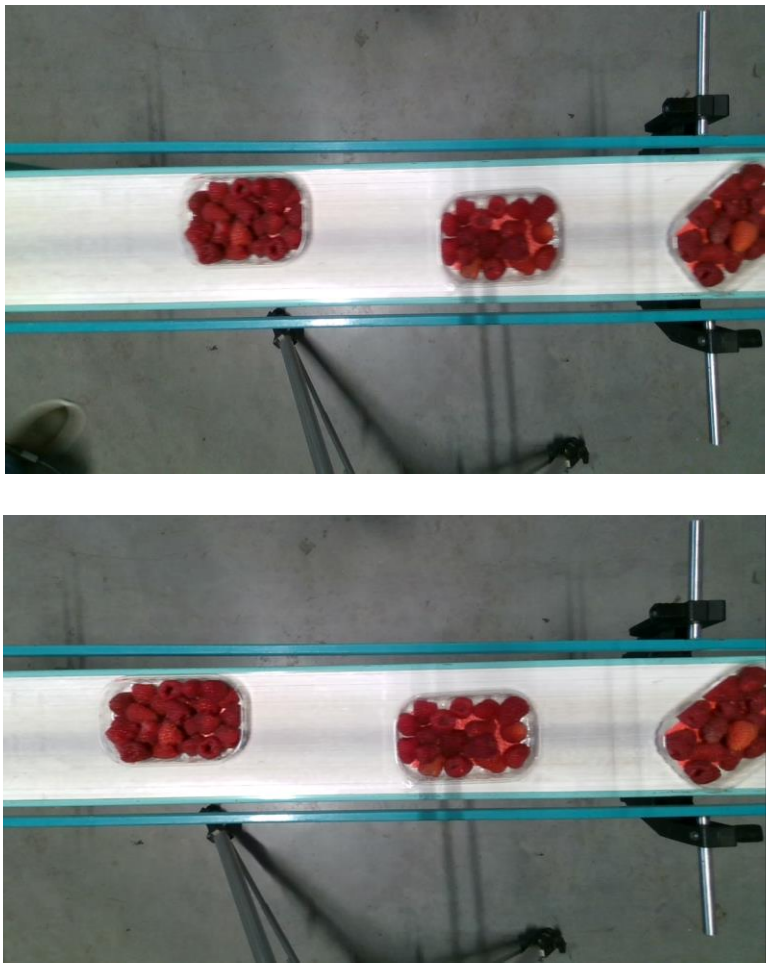

Figure 6: Example of raspberry fruits before (top) and after deblurring (bottom).

In this regard, it is to be noted that the depth acquisition in RealSense™ cannot be carried out beyond a certain depth threshold (about 60 cm), which is the distance at which the RealSense™ sensor was placed. This implies that, while the depth map at such a distance can be acquired properly, the RGB resolution may not be high enough to do the grading task. Indeed, this was confirmed by Sant'Orsola experts who take charge of the annotation (i.e., the annotation involves both the contour of each single fruit and its grading class).

In particular, because of the low RGB resolution and the blurring effect (see the top example of Figure 6), the experts were not able to do the annotation at the fruit level. To mitigate this, the use of state-of-the-art image deblurring solutions [14]. We display a deblurring example in Figure 6, where this blurring effect was alleviated. However, the problem is still not entirely solved, as the colour of the fruits is not representative enough of the five grades due to the low RGB resolution, as confirmed by the experts.

As per the stereo vision images demonstrated in the next sections (**Task 4.3**), the experts also confirmed that the image quality is inadequate to address the annotation and grading parts.

2.2.1.3. Produmar

Before delving into details, we first describe the current operational chain in this pilot. First, an operator picks a whole fish (with the head already trimmed off) and slices it manually into steaks (see Figure 7). Second, the steaks are placed into crates and then stored in a cold chamber. Afterwards, the steaks/fillets are passed onto a conveyor and manually lined up for packaging.

What **AGILEHAND** envisions in this pilot is to automate three tasks. First, the slicing of the whole fish, where cutting lines will be automatically determined. Second, once the fish is sliced into steaks, these should be graded into two grades based on their size (e.g., based on the perimeter of the steak) so that each grade will be placed into a separate crate before being stored in the cold chamber. Third, the steaks/fillets will be automatically picked up by a robotic arm and lined up for packaging. Therefore, the latter task will comprise a vision system to determine the location as well as the depth of the steaks/fillets to be packaged and pass this information to the robotic system for handling.

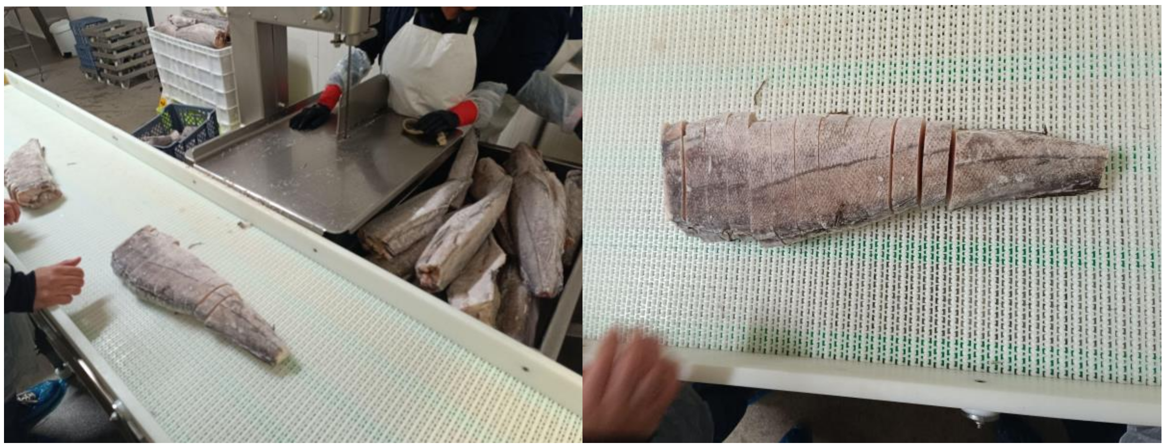
Figure 7: Manual fish slicing example.

To address the aforementioned tasks, a requirement in this use case is to determine precisely, in the acquired images, the pixels that belong to each instance of a certain product (e.g., whole fish, fish fillet, fish steak). This is a computer vision task well known as instance segmentation. Due to the remarkable success of DL in this task [15], in **AGILEHAND,** a recent instance segmentation model has been explored, namely YOLOv8 [16], which is a cutting-edge state-of-the-art instance segmentation and object detection model that introduces impressive improvements concerning prior models in terms of accuracy and speed. In particular, the model was trained on 80% of the acquired data and tested on the remaining 20%. The training was conducted on images with full resolution for 20 epochs. For both tasks, the Mean Average Precision (mAP) of the segmentation model amounts to 99.5%, which is an expected value, given that the task is fairly easy because of the high contrast between the fish and the background. **Figure 8** displays qualitative examples, including both the segment and the bounding box of each instance. The numbers at the corner of the bounding boxes indicate the detection confidence (between 0 and 1).

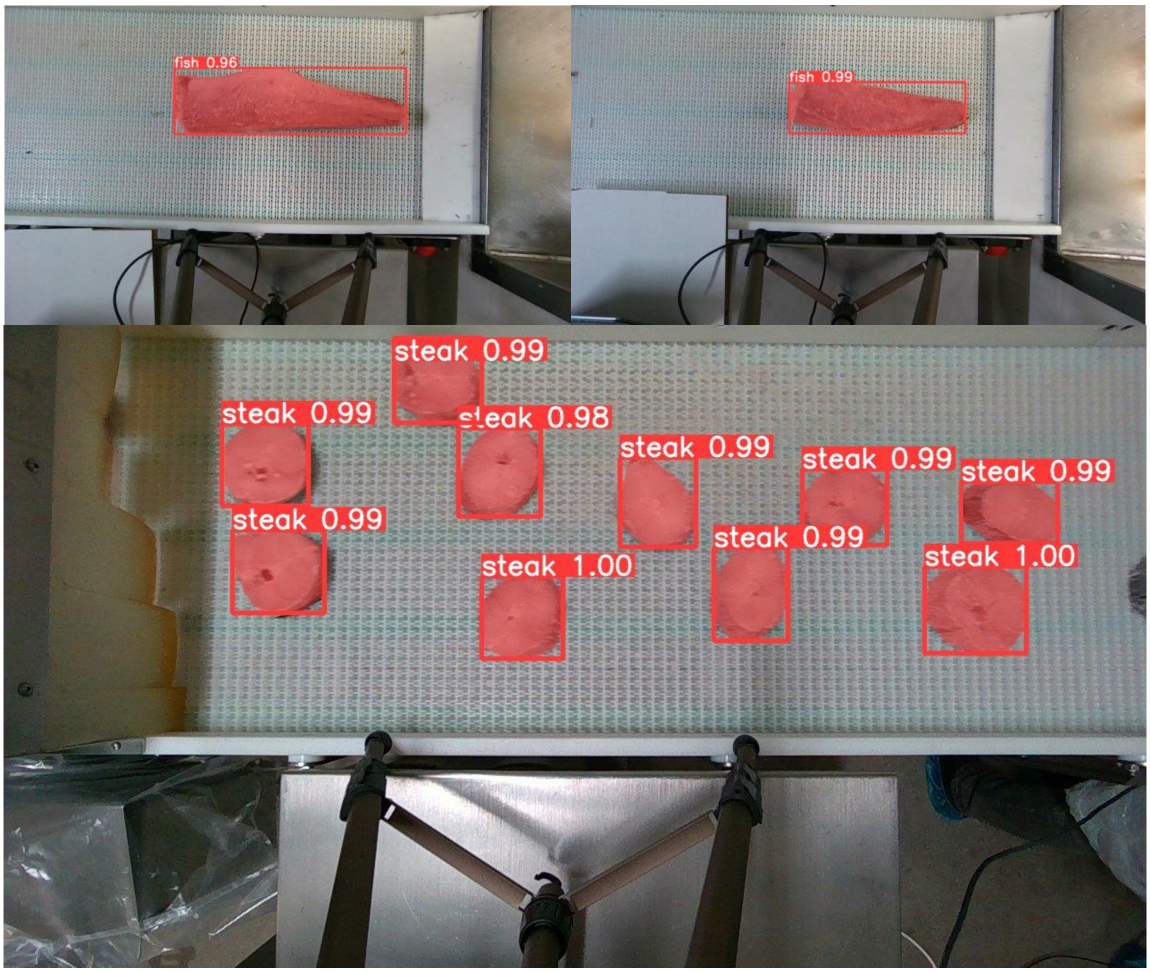

Figure 8: Instance segmentation inference examples. Top: two instances from the whole fish slicing task. Bottom: several instances from the steak grading task. The segmentation confidence (on a scale from 0 to 1) is reported at the top of each instance.

2.2.1.4. Marelec

In this pilot, the task will consist of grading chicken fillets into two classes (A or B) based on the presence of blood clots and rips on the surface of the fillet. To better explain the difference between the A and B grades, some examples are illustrated in Figure 9.

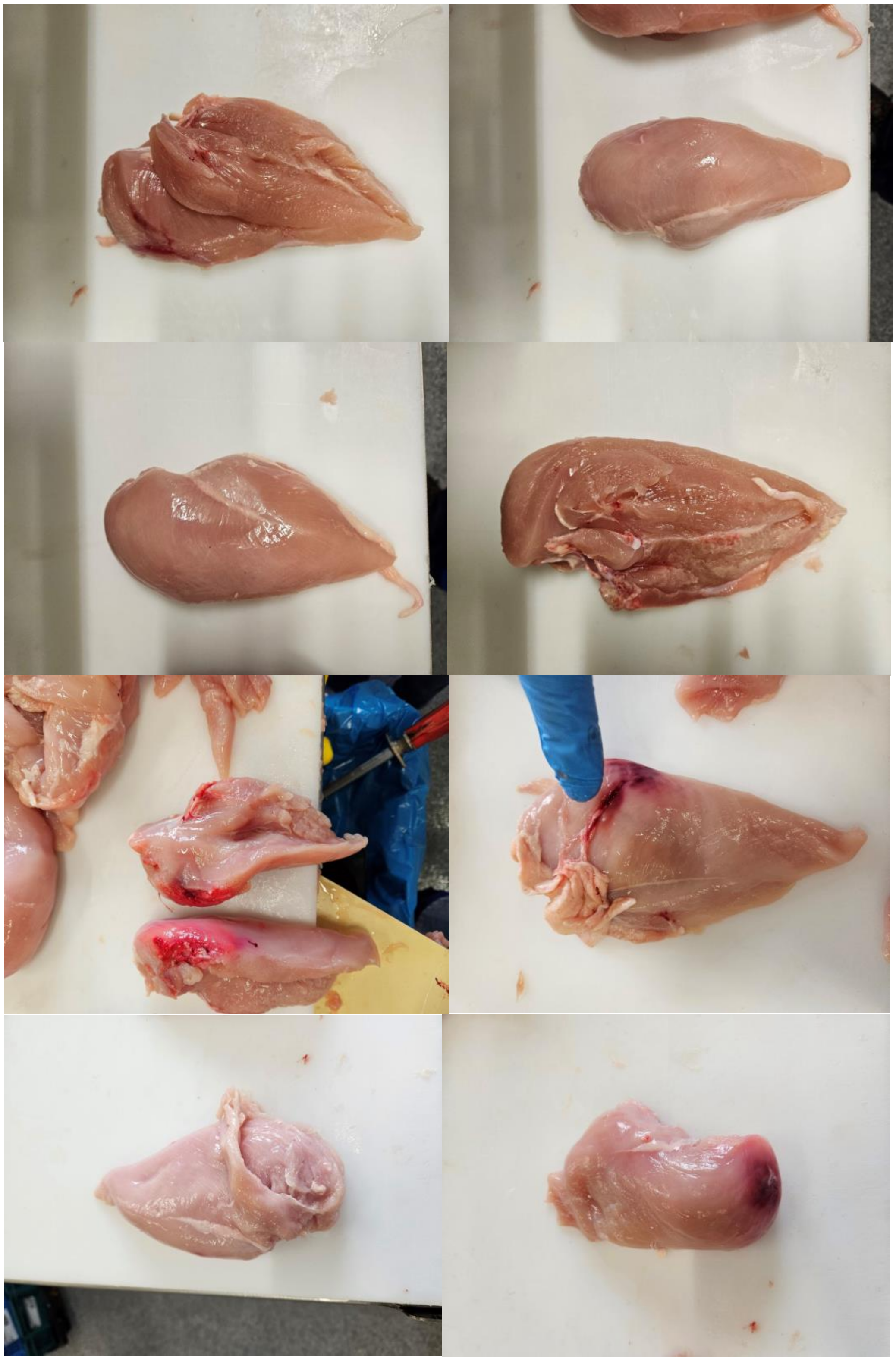

Figure 9: Examples (grabbed by phone) from grades A and B of the Marelec pilot. The top two rows depict A instances, and the bottom two rows illustrate B-grade samples.

It is mentioned that the blood clots may show up on the top and/or the bottom and/or the side of the fillet. Therefore, for each fillet, bottom and top imaging are required for the grading task. Moreover, the segment and the coordinates of the fillets should be communicated to the robotic solution of **AGILEHAND** for further packaging. This suggests the adoption of instance segmentation via the aforementioned YOLOv8 model to segment and grade the fillets.

2.2.2. Future Developments

Regarding **Multiscan**, in the next step, we seek to tune the grading parameters of the machine according to the input batch of oranges to be graded. For instance, the aim is to infer the distribution (e.g., histogram) of the fruits of the input batch based on several criteria such as shape and size.

For the **Sant'Orsola** situation and as mentioned above, the RGB and the depth elements are essential in this use case. The current focus considers the adoption of a LIDAR sensor at the beginning of the conveyor belt. Thus, once the raspberry punnets pass underneath the sensor, they are line-scanned, and the measured depth lines are then put together to build a depth map of each punnet. This enables the adoption of a separate RGB sensor that can be placed close to the punnets for high-resolution imaging, providing a better RGB image quality yet a precise annotation and successive grading as well. In this respect, a further alignment between the RGB images and the depth map is necessary. Once another dataset is acquired by such sensors, a quantitative and qualitative study will be conducted.

The **Produmar** annotation task is still in progress. Precisely, we are working on (i) the annotation of the cutting lines in collaboration with the operator who currently does this task, (ii) the annotation of the packaging datasets (steaks and fillets). Once finalised, the solutions will be developed and a qualitative and quantitative analysis will take place.

Finally, in **Marelec**, the next plan considers the training of YOLOv8 for instance segmentation (and grading) of the chicken fillets and conducting a quantitative and qualitative analysis.

3. AGILEHAND Data-Sets

3.1. Overview

As aforementioned, in **AGILEHAND** we opt for DL methodologies to carry out the instance segmentation, detection and classification tasks required for the machine vision aspect of the project. Therefore, for each use case, **AGILEHAND** requires the acquisition of datasets to enable the training and deployment of the DL models. This section includes a description of the acquisition setups that have been developed so far to explore the use cases and the projected plans regarding the use cases that still require further dataset acquisition campaigns. The work described in this section concerns **Task 4.3**.

3.2. Implementation Status

This section demonstrates how the datasets were acquired in the different pilots, so they can be used in **Task 4.2**.

3.2.1. Current Implementation

3.2.1.1. Multiscan

The dataset was acquired by Multiscan Technologies, S.L (Pol. Ind. Els Algars C/ La Safor, 2 03820 Cocentaina, Alicante – Spain). The dataset contains the images along with their grade annotations (i.e., three quality grades are envisioned, namely Good, Bad, or Undefined). To this end, the oranges go through a roller conveyor that moves them forward and rotates them simultaneously (See Figure 10). As depicted in Figure 11, each orange fruit is captured from different viewpoints. The main factor characterising each of the three classes is the presence as well as the size of bruises and blemishes on the skin of the oranges. However, since oranges are spherical objects, it is necessary to cover the different sides of a given orange and we call this 'a multiview analysis' as opposed to a 'single view analysis' in which only one side of the orange is acquired by the cameras. In further detail, to ensure better visibility of the oranges, multiple views of each orange are combined to form one single RGB image, as shown in Figure 11.

The oranges were captured using a Sony IMX429 camera placed in a top-down view from approximately 1 metre from the fruit plane. The oranges are exposed to Cool white LEDs as a uniform lighting source.

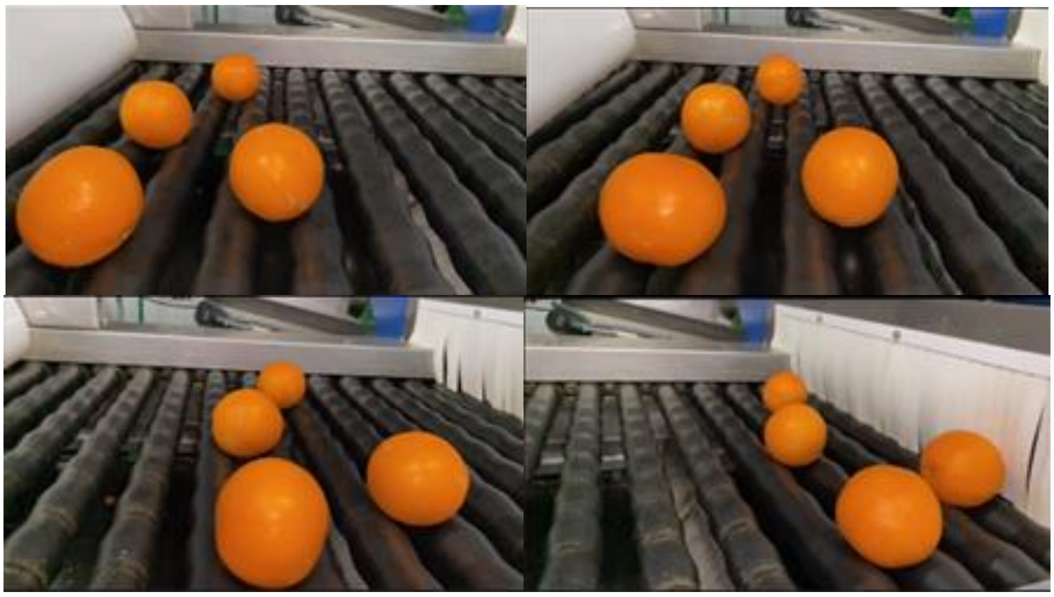
Figure 10: *Several instances of oranges on the roller conveyor.*

Figure 11 also shows instances from the three grading classes. The Good grade oranges manifest clean skin, while the Bad class oranges often contain blemishes and bruises of various severities on the outside. The Undefined class oranges, however, report imperfections that are neither too grave to be categorised as Bad nor insignificant to be considered Good.

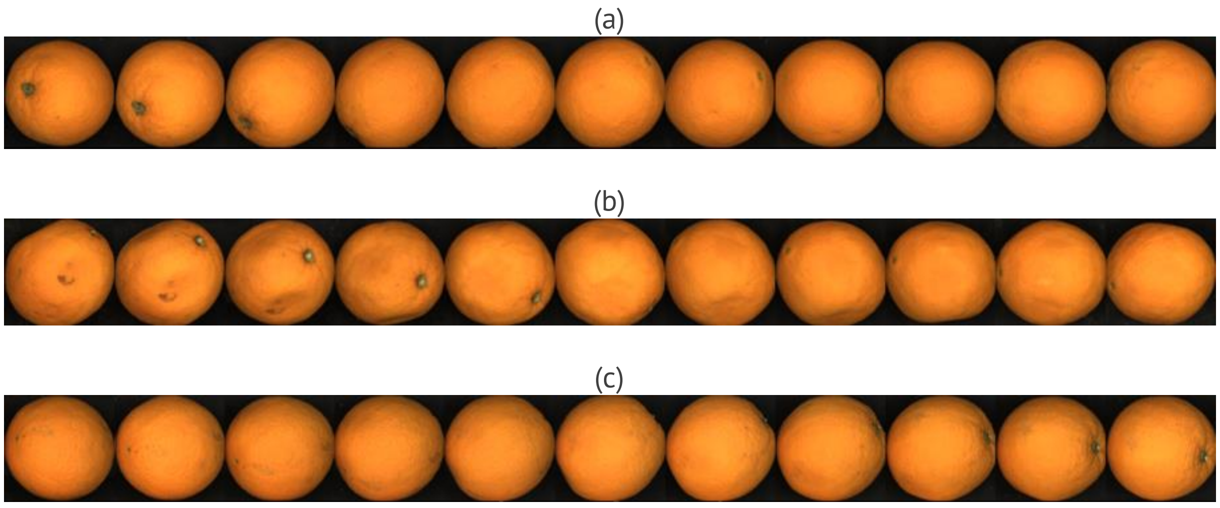

Figure 11: *Samples were viewed from different angles from each class. (a) Good, (b) Bad, and (c) Undefined.*

The dataset totals 452 orange samples (each sample features several views of the same orange) with highly varying sample distribution per class. The dataset can be divided into two subsets, namely a training set of 317 samples (70% of the dataset) and 135 test samples (yet 30% of the total count). The statistics are given in Table 4.

Table 4: Statistics of the orange dataset.

	Training	Test
Good	78	33
Bad	206	88
Undefined	33	14
Total	317	135

3.2.1.2. Sant'Orsola

Due to the particularity of the Raspberry grading task that addresses small fruits, three sensing options have been explored so far. In particular, RealSense™ D415, RealSense™ D456 and a stereo vision system were considered. In all the three experimental setups, the sensor is mounted in a top-down view at the end of a horizontal support that is fixed on a vertical tripod.

While the punnets move along the conveyor, the RealSense™ sensor (D415 or D456) acquires the RGB and depth frames at a frame rate of 30 FPS (frames per second) and communicates them to the computer via USB cable (see Figure 12).

Similarly, the pair of stereo cameras are mounted in the same way. However, the stereo cameras are connected to the computer via an RJ45 cable and acquire the left and right image pairs at 15 FPS (see Figure 13).

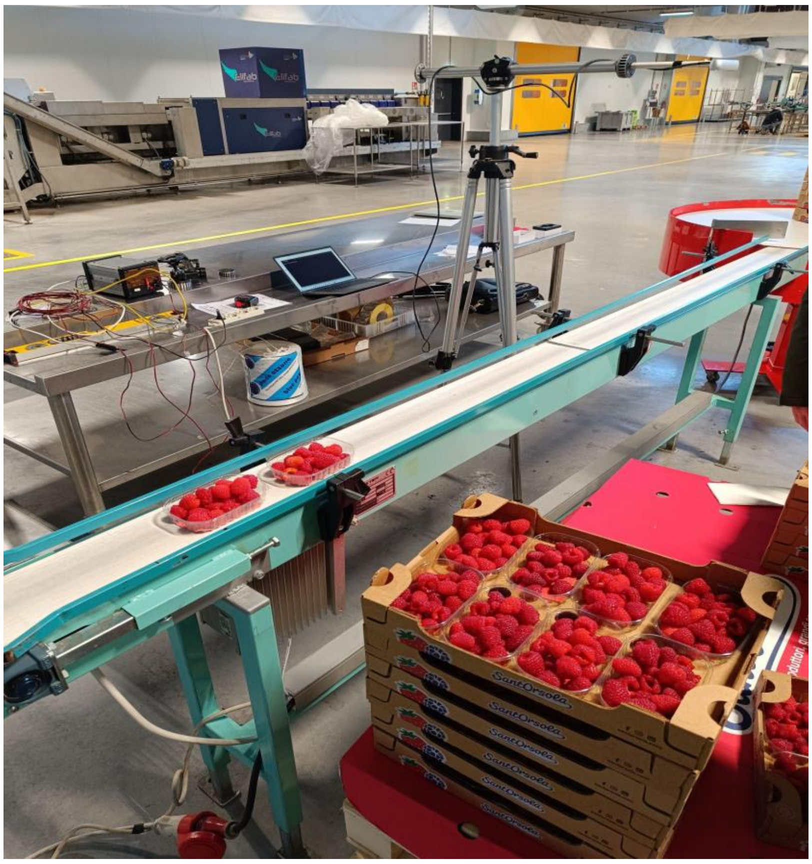

Figure 12: Acquisition setup using Intel® RealSense™ D456 sensor.

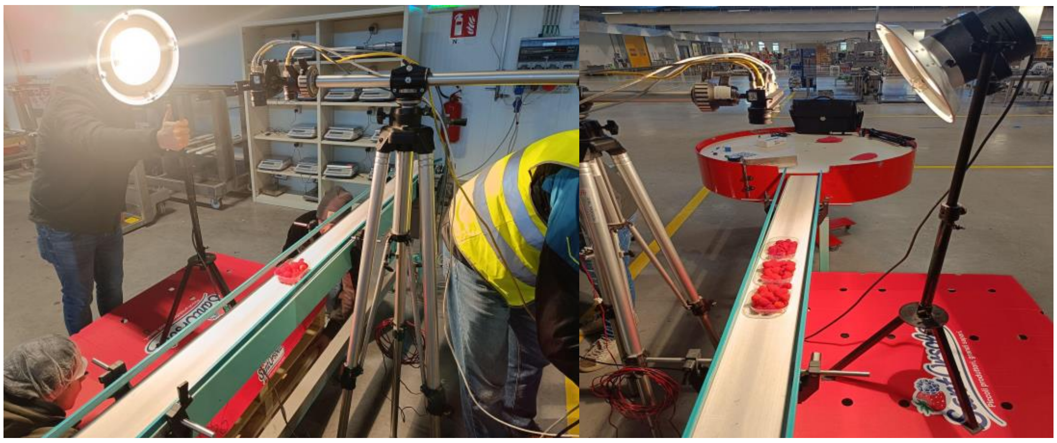

Figure 13: Acquisition setup using the stereo vision system.

Regarding lighting, no illumination source was used regarding the RealSense™ D415 sensor as it was a first feasibility study, given that the lighting conditions at the production line at **Sant'Orsola** were presumably fair. As per the other two sensing solutions, acquisition attempts were made both in ambient lighting conditions as well as with a halogen lamp (see Figure 12). Nevertheless, no differences were observed when using the Realsense D456 sensor, whilst for the stereo system, the artificial illumination was slightly better.

Figure 14**,** Figure 15 and Figure 16 illustrate some acquired samples regarding Intel® RealSense™ D415, D456 and the stereo vision, respectively. Note that the stereo solution implies the acquisition of two frames simultaneously via the left and right cameras.

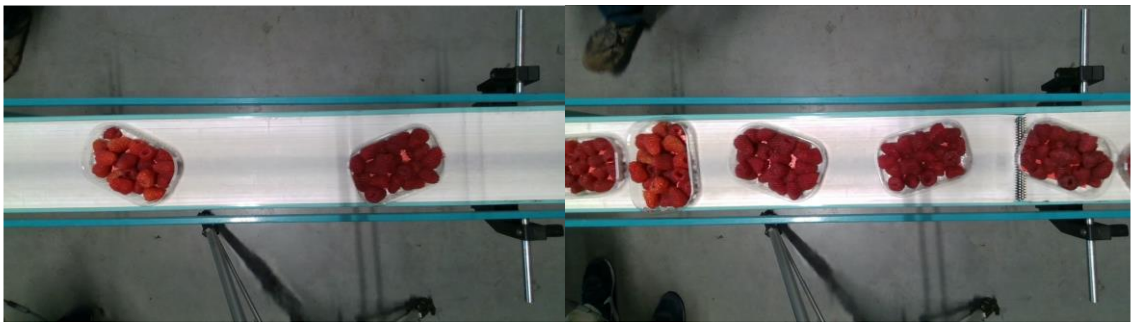

Figure 14: Raspberry instances captured through Intel® RealSense™ D415.

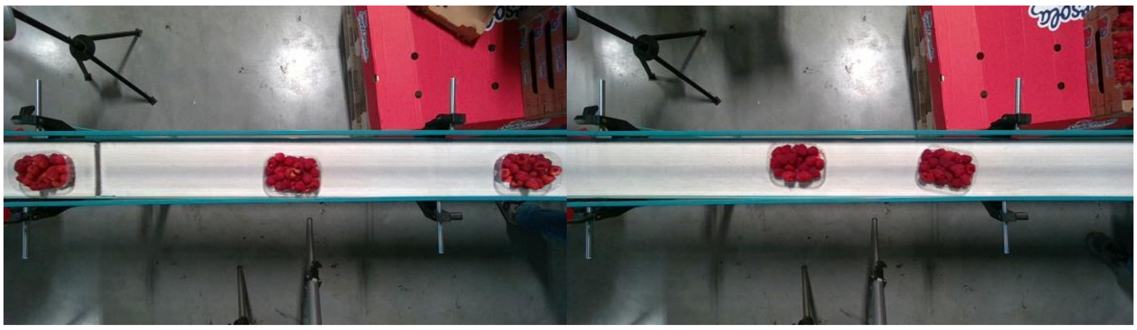

Figure 15: Raspberry instances captured using Intel® RealSense™ D456.

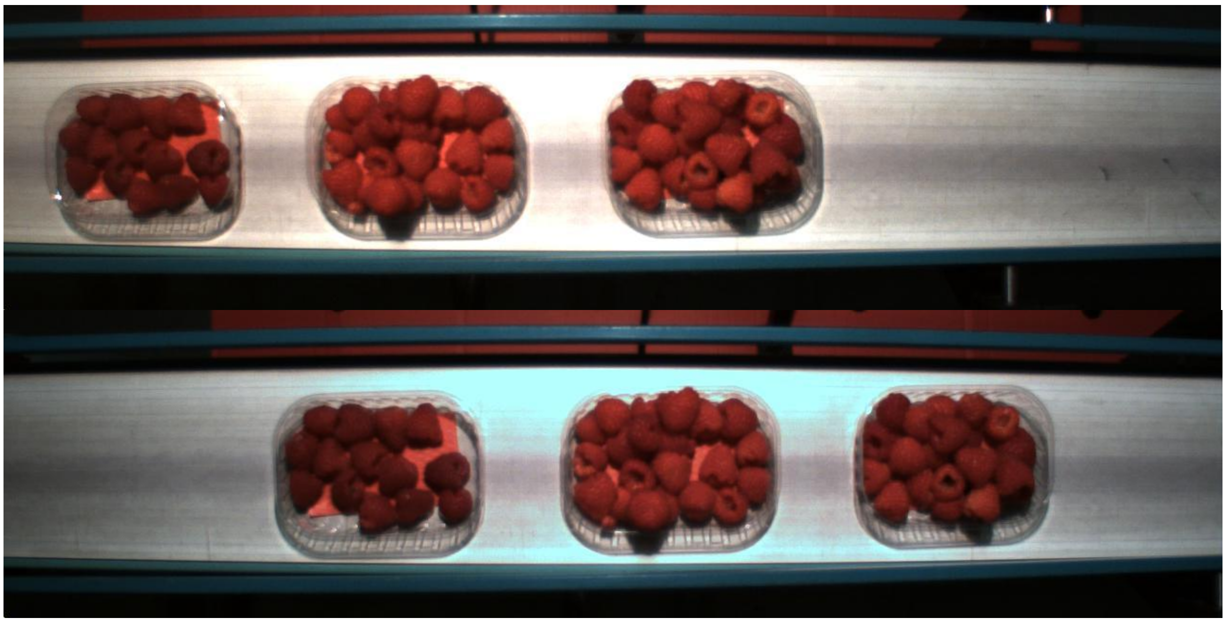
Figure 16: Raspberry instances captured by the stereo cameras (FLIR Blackfly BFLY-PGE-13E4C-CS). Top: left frame. Bottom: Right frame.

3.2.1.3. Produmar

In this pilot, three tasks will be covered in AGILEHAND. In particular, the first task involves slicing a whole fish into steaks, where the scope is to determine the cutting lines starting from the head side of the fish (note that the head of the fish would be already cut off when it arrives at the production line). The second task addresses the problem of steak grading. The third and fourth tasks require the determination of the segment as well as the location of fish steaks and fillets to automate the packaging process, respectively.

In this respect, the same acquisition setup was adopted to capture all the three datasets. Precisely, a similar acquisition system to the one adopted for the use-case of Sant'Orsola with RealSense™ D456 was used as shown in Figure 17. Examples from the acquired datasets for each task are provided in Figure 18, Figure 19 and Figure 20.

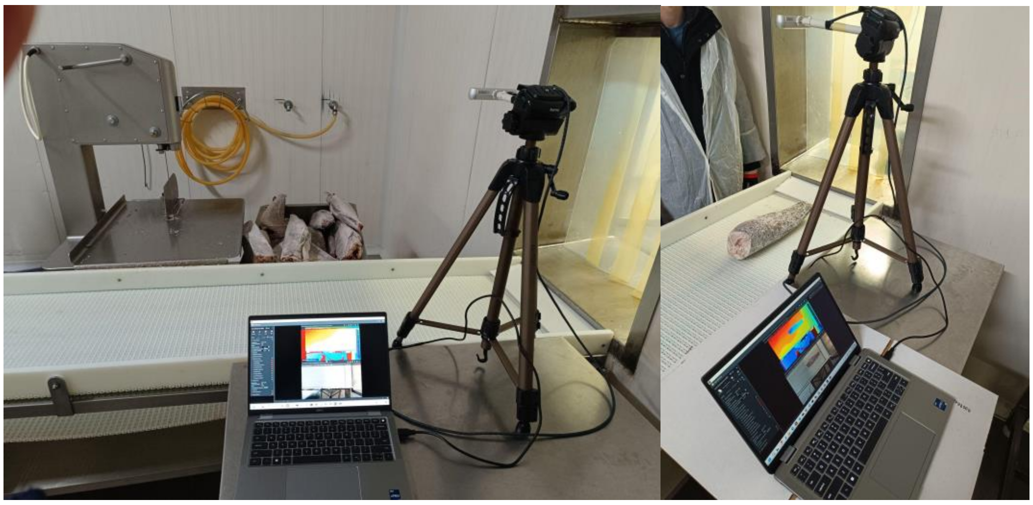
Figure 17: Produmar acquisition system using Intel® RealSense™ D456.

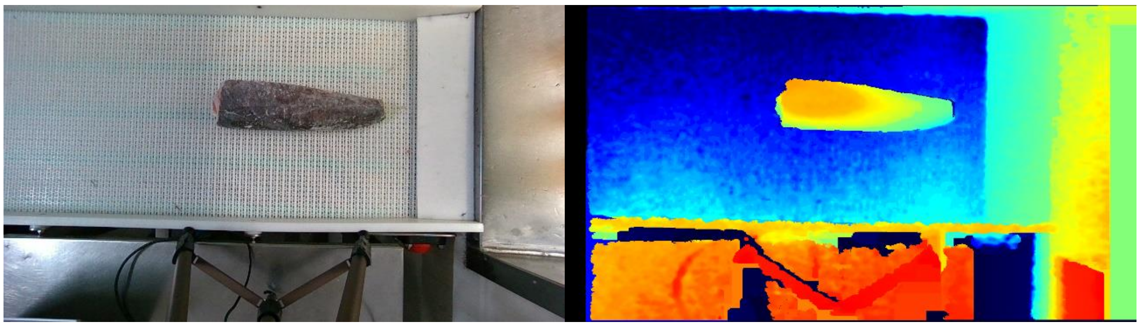

Figure 18: RGB and depth images of an example from the whole fish slicing task.

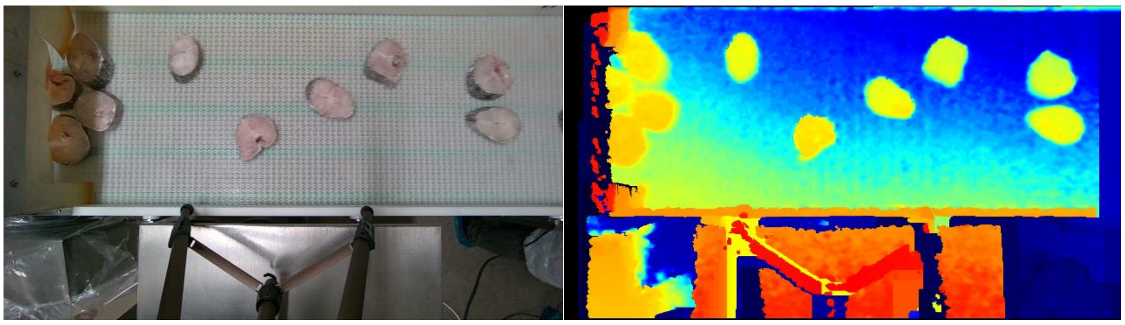

Figure 19: RGB and depth images of an example from the steak classification task.

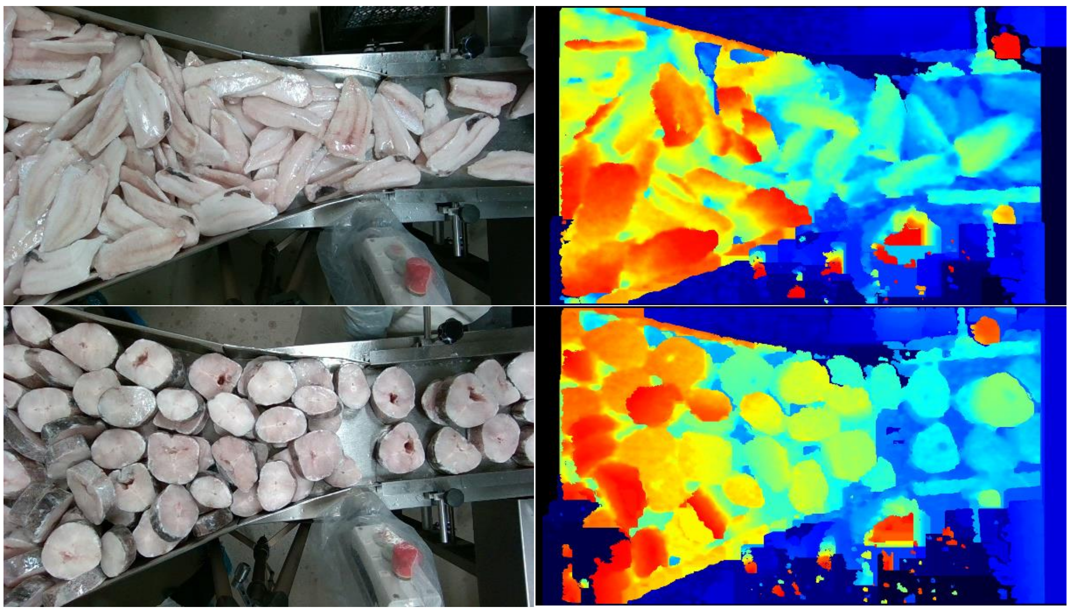

Figure 20: RGB and depth examples from the fillets/steaks packaging task.

The annotation of the datasets has been performed using the well-known CVAT (Computer Vision Annotation Tool) [17], where the contour of each fish sample is manually annotated as depicted in Figure 21.

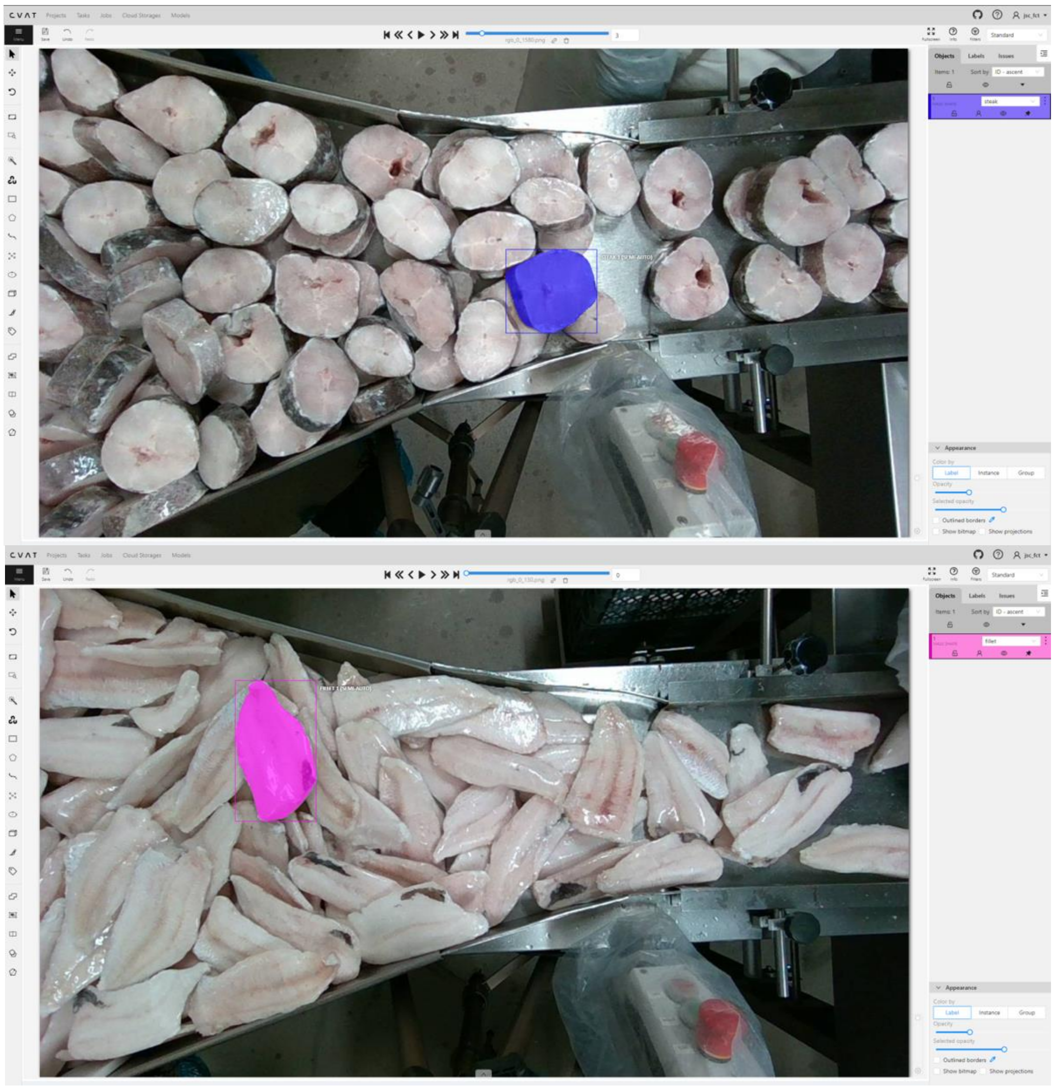

Figure 21: *Steak and fillet manual annotation examples using CVAT.*

For the first task (fish slicing), a total of 617 fish samples were acquired, while for the steak grading task, 103 examples were acquired. For both scenarios, the datasets were split into 80% for training and 20% to test the trained instance segmentation model.

It is to be noted that for the use cases that consider the adoption of Intel® RealSense™, the images were always acquired with the highest resolution (1280x720) to allow the highest possible quality. For the same reason, the sensor was placed at the minimal distance allowed to enable depth measurement (i.e., about 60 cm).

3.2.2. Future Developments

Regarding the first use case with **Multiscan**, the next step is to acquire a dataset of orange batches before they go into the grading machine and the annotation of the acquired images.

As per the second use case with **Sant'Orsola**, currently, better illumination conditions are being explored. For instance, ring lights can be a good option to prevent casting shadows from all directions and to emphasise a beam of uniform lighting on the central part where the punnets of raspberries will be placed. Further, as aforementioned, the plan is to acquire the depth and the RGB images separately to enable higher resolution.

Concerning the third use case with **Produmar**, the packaging datasets are currently being annotated. Further, although the segments of the fish were annotated for the fish-slicing task, the slicing lines and the packaging datasets are still being annotated in collaboration with experts from **Produmar**.

Finally, a chicken fillets dataset will be soon acquired in collaboration with **Marelec**. The annotation will also be carried out using the CVAT tool.

4. AGILEHAND Smart Sensing SUITE: Functional and Implementation Viewpoints

In this section, the overview of the functional and implementation viewpoints of the Smart Sensing SUITE are presented.

4.1.1. Functional Viewpoint

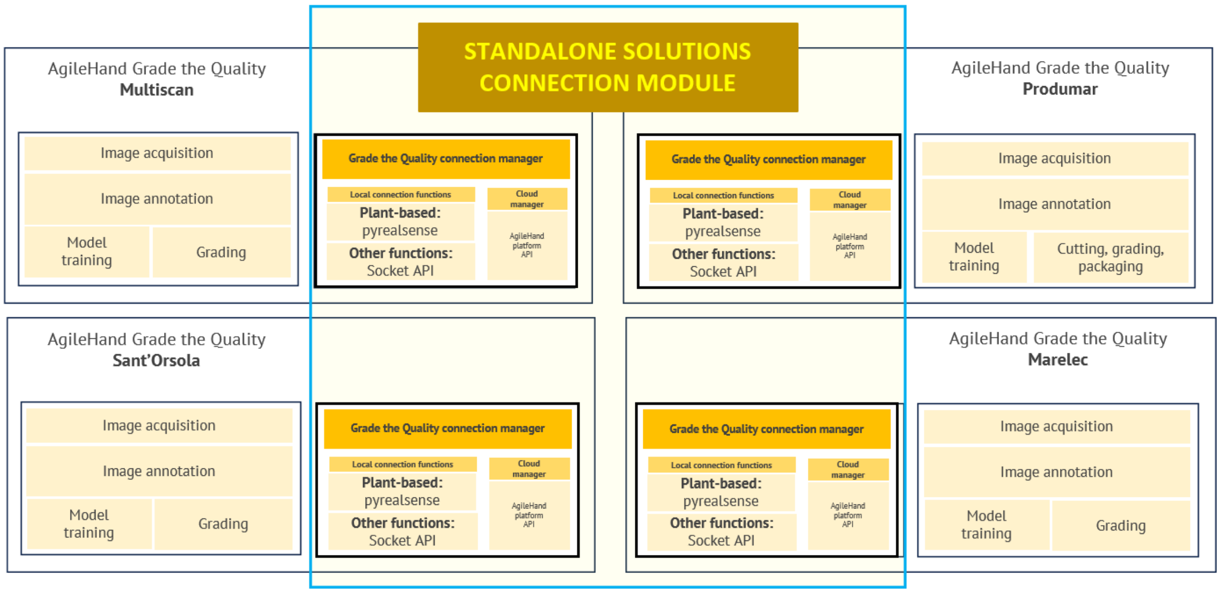
STANDALONE SOLUTIONS CONNECTION MODULE
AgileHand Grade the Quality Multiscan
Image acquisition
Image annotation
Model training
Grading
Grade the Quality connection manager
Local connection functions
Plant-based: pyrealsense
Other functions: Socket API
Cloud manager
AgileHand platform API
AgileHand Grade the Quality Produmar
Image acquisition
Image annotation
Model training
Cutting, grading, packaging
AgileHand Grade the Quality Sant'Orsola
Image acquisition
Image annotation
Model training
Grading
AgileHand Grade the Quality Marelec
Image acquisition
Image annotation
Model training
Grading

Figure 22: Solution Composition.

The functional viewpoint of the Smart Sensing Suite is presented in **Figure 22**. The Smart Sensing Suite uses the images captured by the sensors defined in **Task 4.1** (Intel RealSense™ D415, Intel RealSense™ D456 and Stereo Vision (FLIR Blackfly BFLY-PGE-13E4C-CS)), which are annotated in **Task 4.3**, with the necessary information for supporting the training of the AI models to be used in the Smart Grading solution developed in **Task 4.2**.

4.1.1.1. Data Structure of Smart Sensing Suite

Table 5: Input / Output Data Format.

Format	Input/Output	Example
.PNG	Input	Acquired images.
.json	Input	Annotation information (e.g., coordinates of the pixels in the image that pertain to a certain fruit, the quality grade of the fruit)
.json/.txt	Output	Coordinates of the pixels that belong to the instance segmentation masks of different products (e.g., fruit, fish).

4.1.1.2. Smart Sensing Suite Software Requirements

Table 6: Software requirements.

Software Component	Description/Role	Required Version/Configuration	Dependencies
Windows OS	Operating system needed to use the tool	Windows 10 Pro	N/A
Python	Programming language	Latest	N/A
Anaconda	Distribution of Python meant to simplify package management and deployment	Latest	N/A

4.1.1.3. Objects

- **Frontend**: Represents the user interface and the presentation layer.
  - Native User Interface: create a graphical user interface with menus that are both functional and user-friendly. Specifically, the user will be able to run product grading tasks and customize different parameters according to the grading criteria.
- **Backend**: Represents the application logic and the server layer.
  - Native Solution Component
    - Core Functions: Product detection, segmentation and grading. Further results visualization can also be implemented for qualitative analysis.
    - Connection Manager: Network socket.

4.1.2. Implementation Viewpoint

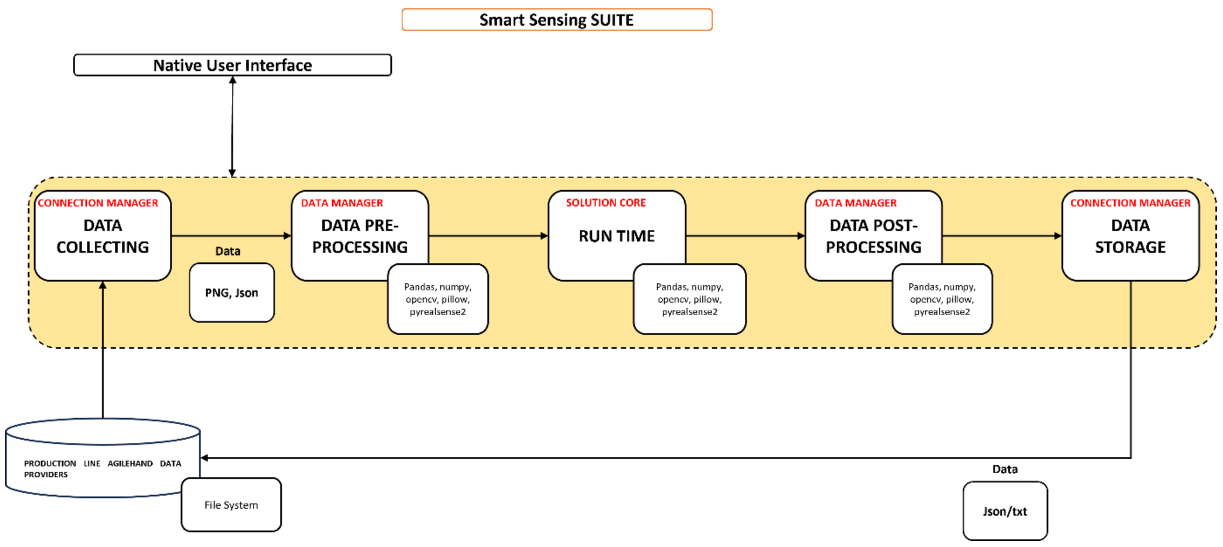

Figure 23: Implementation Architecture.

The implementation architecture represented in Figure 23 presents an overview of the implementation components detailed in the following tables.

4.1.2.1. *Implementation Components*

Table 7: Implementation Components.

Implementation Components	Description
Data Collecting	Serves for visualizing the RGB and depth images before, during and after acquisition.
Data Pre-Processing	Serves as an image preprocessing step to enable further processing stages. For instance, image noise removal and hole-filling operations are applied.
Run Time	Serves for deploying the developed grading solutions. It also offers quality check modules to ensure that the data acquisition is smooth.
Data Post-Processing	Serves for finetuning the obtained grading results and removing outliers. For instance, when instance segmentation includes irrelevant objects.
Data Storage	Serves for storing qualitative and quantitative outputs of the grading solutions.

4.1.2.2. *Technical Description of Smart Sensing Suite Components*

Following the identification and specification of the implementation components in the previous section, the technical description of each component is provided in the following tables. It details the set of technologies required to implement each of the implementation components.

Table 8: Technical Description of Data Collecting Implementation Component

Implementation component	Data Collecting
Description of the implementation component	Serves for visualizing the RGB and depth images before, during and after acquisition.
Used technologies	Python, Intel® RealSense™.
Technical Description of the Component	Dependencies
	Development Language: Python. Libraries: Pandas, numpy, opencv, pillow, pyrealsense2.
	Interfaces
	User Interface: Intel® RealSense™ Viewer. Network/Protocols: HTTP/HTTPS.

Table 9: Technical Description of Data Pre-Processing Implementation Component

Implementation component	Data Pre-Processing
Description of implementation component	Serves as an image preprocessing step to enable further processing stages. For instance, image noise removal and hole filling operations are applied.
Used technologies	Python, Intel® RealSense™.
Technical Description of the Component	Dependencies
	Development Language: Python. Libraries: Pandas, numpy, opencv, pillow, pyrealsense2.
	Interfaces
	User Interface: Intel® RealSense™ Viewer. Network/Protocols: HTTP/HTTPS.

Table 10: Technical Description of Run Time Implementation Component

Implementation component	Run Time
Description of implementation component	Serves for deploying the developed grading solutions. It also offers quality check modules to ensure that the data acquisition is smooth.
Used technologies	Python, Intel® RealSense™.
Technical Description of the Component	Dependencies
	Development Language: Python Libraries: Pandas, numpy, opencv, pillow, pyrealsense2.
	Interfaces
	User Interface: Intel® RealSense™ Viewer. Network/Protocols: HTTP/HTTPS.

Table 11: Technical Description of Data Post-Processing Implementation Component

Implementation component	Data Post-Processing
Description of implementation component	Serves for finetuning the obtained grading results and remove outliers. For instance, when instance segmentation includes irrelevant objects.
Used technologies	Python.
Technical Description of the Component	Dependencies
	Development Language: Python Libraries: Pandas, numpy, opencv, pillow, pyrealsense2.
	Interfaces
	User Interface: Intel® RealSense™ Viewer. Network/Protocols: HTTP/HTTPS.

Table 12: Technical Description of Data Storage Implementation Component

Implementation component	**Data Storage**
Description of implementation component	Serves for storing qualitative and quantitative outputs of the grading solutions.
Used technologies	Python
Technical Description of the Component	Dependencies
	Development Language: Python. Libraries: Json.

5. Conclusion

This document described the development progress within WP4 of **AGILEHAND**. While most use cases can be addressed relatively well in terms of sensing and grading precision, the use case being addressed with **Sant'Orsola** requires delicate handling due to several challenges such as (i) the size of the rather small fruits; (ii) the occlusion, which multiplies the difficulty of the problem, as some fruits may often show up partially; and (iii) the motion of the punnets, which complicates further the task. Current efforts are emphasized on handling the RGB and depth acquisition for this use case to acquire high-quality images that enable precise grading outcomes. As per the other use cases, the grading concerns large objects (e.g., fish and chicken fillets), yet they pose limited challenges that can be tackled.

References

[1] Li, X., & Zhu, W. (2011). Apple grading method based on features fusion of size, shape and color. Procedia Engineering, 15, 2885-2891.

[2] Xu, B., Cui, X., Ji, W., Yuan, H., & Wang, J. (2023). Apple grading method design and implementation for automatic grader based on improved YOLOv5. Agriculture, 13(1), 124.

[3] Jayasundara, J. M. V. D. B., Ramanayake, R. M. L. S., Senarath, H. M. N. B., Herath, H. M. S. L., Godaliyadda, G. M. R. I., Ekanayake, M. P. B., ... & Ariyawansa, S. (2023). Deep learning for automated fish grading. Journal of Agriculture and Food Research, 14, 100711.

[4] Wold, J. P., Veiseth-Kent, E., Høst, V., & Løvland, A. (2017). Rapid on-line detection and grading of wooden breast myopathy in chicken fillets by near-infrared spectroscopy. PLoS One, 12(3), e0173384.

[5] Liming, X., & Yanchao, Z. (2010). Automated strawberry grading system based on image processing. Computers and electronics in agriculture, 71, S32-S39.

[6] Behera, S. K., Jena, L., Rath, A. K., & Sethy, P. K. (2018, April). Disease classification and grading of orange using machine learning and fuzzy logic. In 2018 International Conference on Communication and Signal Processing (ICCSP) (pp. 0678-0682). IEEE.

[7] Dong, S., Wang, P., & Abbas, K. (2021). A survey on deep learning and its applications. Computer Science Review, 40, 100379.

[8] Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. Computers and electronics in agriculture, 147, 70-90.

[9] Niu, S., Liu, Y., Wang, J., & Song, H. (2020). A decade survey of transfer learning (2010–2020). IEEE Transactions on Artificial Intelligence, 1(2), 151-166.

[10] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). Ieee.

[11] Wu, Y., Chen, Y., Yuan, L., Liu, Z., Wang, L., Li, H., & Fu, Y. (2020). Rethinking classification and localization for object detection. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 10186-10195).

[12] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[13] Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. arXiv preprint arXiv:1602.07360.

[14] https://github.com/zzh-tech/ESTRNN

[15] Gu, W., Bai, S., & Kong, L. (2022). A review on 2D instance segmentation based on deep neural networks. Image and Vision Computing, 120, 104401.

[16] https://github.com/ultralytics/ultralytics

[17] Accessible at: https://app.cvat.ai/