FLEXIGROBOTS

D3.2 FlexiGroBots Platform v2

Document Identification			
Status	Final	**Due Date**	31/12/2022
Version	1.0	**Submission Date**	10/03/2023

Related WP	WP3	**Document Reference**	D3.2
Related Deliverable(s)	D2.3, D3.1, D3.3	**Dissemination Level (*)**	PU
Lead Participant	ATOS	**Lead Author**	Sergio García, ATOS
Contributors	SER, CSIC, WUR, VTT, BIO, ART, LUKE, IDSA	**Reviewers**	Ismael Suárez Cerezo, SER
			Ángela Ribeiro, CSIC

Co-funded by the Horizon 2020 Programme of the European Union

Document Information

List of Contributors	
Name	Partner
Sergio García	ATOS
Mario Triviño	ATOS
Miguel A. Esbrí	ATOS
Daniel Rodera	ATOS
A. Carlos Cob Parro	ATOS
Daniel Calvo	ATOS
Mar Ariza Sentís	WU
Sergio Vélez	WU
João Valente	WU
Marko Panić	BIO
Oskar Marko	BIO
Juha-Pekka Soininen	VTT
Kari Kolehmainen	VTT
Ángela Ribeiro	CSIC
Álvaro López	CSIC
Artur Bogucki	CEPS
Moritz Laurer	CEPS

Document History			
Version	Date	Change editors	Changes
0.1	07/11/2022	Sergio García (ATOS)	Initial document outline
0.2	18/11/2022	Miguel A. Esbrí (ATOS), Daniel Rodera (ATOS)	Section 4
0.3	23/11/2022	A. Carlos Cob (ATOS)	Section 3 and section 2
0.4	02/12/2022	Mario Triviño (ATOS)	Section 5
0.5	05/12/2022	Sergio García (ATOS)	Section 1

Document History			
Version	Date	Change editors	Changes
0.6	05/12/2022	Marko Panić (BIO), Oskar Marko (BIO)	Section 2
0.7	06/12/2022	Daniel Calvo (ATOS)	Section 1.2
0.8	07/12/2022	Juha-Pekka Soininen (VTT), Kari Kolehmainen (VTT)	Section 6
0.9	07/12/2022	Sergio García (ATOS)	Conclusions
0.10	12/12/2022	Mario Trivino (ATOS)	Section 5.2.1
0.11	12/12/2022	Angela Ribeiro (CSIC)	Complete section 6 and include Annex B
0.11	13/12/2022	Juha-Pekka Soininen (VTT)	Complete section 6
0.12	13/12/2022	Angela Ribeiro (CSIC)	Adjustment of content Annex B
0.13	14/12/2022	Sergio García (ATOS)	Formatting
0.14	16/12/2022	Sergio García (ATOS)	Styling improvements
0.15	24/01/2023	Artur Bogucki (CEPS), Moritz Laurer (CEPS)	Section 1.4.3
0.16	26/01/2023	Juha-Pekka Soininen (VTT)	Minor additions to section 6
0.17	31/01/2023	Oskar Marko (BIO)	Minor additions to section 2
0.18	01/02/2023	A. Carlos Cob (ATOS)	Additions to section 3
0.19	01/02/2023	Mario Trivino (ATOS)	Sections 1.4.1.2 and 1.4.2
0.20	06/02/2023	Sergio García (ATOS)	Major additions to section 1 and 2.3
0.21	10/02/2023	Angela Ribeiro (CSIC)	Minor additions to section 6 and Annex B
0.22	20/02/2023	Mario Trivino (ATOS)	Annex A
1.0	01/03/2023	Sergio García (ATOS)	FINAL VERSION TO BE SUBMITTED

Quality Control		
Role	Who	Approval Date
Deliverable leader	Sergio García, ATOS	01/03/2023
Quality manager	Ivan Zaldivar Santamaria, ATOS	09/03/2023
Project Coordinator	Francisco Javier Nieto de Santos, ATOS	10/03/2023

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation / acronym	Description
ADS	Agriculture Data Space (ADS)
AI	Artificial Intelligence
AWS	Amazon Web Services
CVRP	Capacitated Vehicle Routing Problem
DL	Deep Learning
DS	Data Space
GDAL	Geospatial Data Abstraction Library
GeoTiff	Georeferenced TIFF
GPU	Graphics Processing Unit
IDSA	International Data Space Association
JSON	JavaScript Object Notation
MCC	Mission Control Centre
ML	Machine Learning
mTSP	Multi Traveling Salesman Problem
NSGA-II	Non-dominated Sorting Genetic Algorithm II
ODC	Open Data Cube
OGC	Open Geospatial Consortium
QGIS	QGIS is a free and open-source cross-platform desktop geographic information system application that supports viewing, editing, printing, and analysis of geospatial data
STAC	Spatio-Temporal Asset Catalogue
TSP	Traveling Salesman Problem
TLS	Transport Layer Security
UAV	Unmanned Aerial Vehicle
VM	Virtual Machine
YAML	YAML Ain't Markup Language
WP	Work Package

Executive Summary

The purpose of deliverable D3.2 is to collect the progress made in FlexiGroBots' WP3 (Platform development) during the last months of work –specifically from January 2022 (M13) to December 2022 (M24). This is the second out of three deliverables corresponding to this work package. D3.3 will be published in December 2023 (M36), at the end of the project.

The concepts underlying this WP remain the same as those included in D3.1 [1] –please refer this document for the complete content. In short, *"The FlexiGroBots platform is devoted to enabling the usage of flexible and heterogeneous multi-robot systems for intelligent automation of precision agriculture operations. It implements:*

- *An Agriculture Data Space (ADS) based on the existing solution from the International Data Spaces Association (IDSA). The ADS will enable data sharing across pilots ensuring sovereignty, governance, and security.*
- *The components supporting Machine Learning Operations (MLOps), addressing the complete lifecycle of the ML models.*
- *The enablers to deploy geospatial data management, access and processing capabilities following Open Geospatial Consortium's (OGC) standards.*
- *The common and general applications and services, implemented with multiple purposes to be used off the shelf by more new farmers.*
- *The Mission Control Centre (MCC), a solution to plan, execute and monitor the operation of fleets of flexible and heterogeneous robots –providing the maximum standards in terms of safety– integrated with the rest of the components of the platform through the ADS."*

This document describes thoroughly the status of these components at M24 of the project following the same document structure used for D3.1. This will make it easier for the reader to understand the progress, improvements and changes carried out within the project since the first version of this deliverable (M12). The advances corresponding to the last year of the project that are included in this document are also supported by the progress of the developments in the project's repository: https://github.com/FlexiGroBots-H2020

The main conclusions of the document are that some of the main components of the platform are deployed –including the Artificial Intelligence prototype, the Data Space, and the geospatial data management platform– as well as some of the core components of the Mission Control Centre. Additionally, common applications for different purposes have been developed on top of, mainly, the Artificial Intelligence platform. Moreover, this second version of the deliverable also details those developments that have not been successfully completed within the time allotted to some tasks of the work package. For this reason, updated workplans are established, not only for those tasks whose duration extends to the third year of the project, but also for those that have not been completed.

1 Introduction

1.1 Purpose of the document

The goal of this document is to describe the FlexiGroBots platform at the end of the second year of the project. For the sake of brevity and since this document is the second iteration of D3.1, the same statements will not be repeated here –please refer D3.1 for the complete description of document's purpose. In summary, the FlexiGroBots platform is made up of the following components:

- Artificial Intelligence (AI) platform.
- Common data enablers and services.
- Geospatial enablers and services.
- Common application services.
- Mission Control Centre.

The status of the developments corresponding to each of these components is described in this document following the structure that was already introduced in the previous version of the deliverable (D3.1):

- Implemented functionalities.
- Requirements: technical and functional.
- Data models.
- Application Programming Interfaces (APIs).
- Graphical User Interfaces (GUIs).
- Installation procedure.
- Prototype availability within FlexiGroBots.
- Release planning.

It is worth noting that the effort put into the project during the last year enables now to include more information on aspects that were not sufficiently consolidated in the previous version of this document, such as data models for some of the components.

Other more general aspects of the work package have not changed either:

- The Agile methodology has been continued for the planning and monitoring of the tasks –*user stories*– that allow the different developments to be carried out. The status of these tasks is reviewed in monthly retrospective meetings –according to *sprints* length.
- The use of open-source code both for platform components –those relying on existing software– and for the outcomes of the project.

Please refer D3.1 if further information is required regarding any of the previous topics.

1.2 Relation to other project activities

Deliverable D3.2 is the interim outcome of WP3 - *Platform development*, wrapping the results of several tasks: *T3.1 - AI platform*, *T3.2 - Common data enablers and services*, *T3.3 - Geospatial enablers and services*, *T3.4 - Common application services* and *T3.5 - Mission Control Centre*.

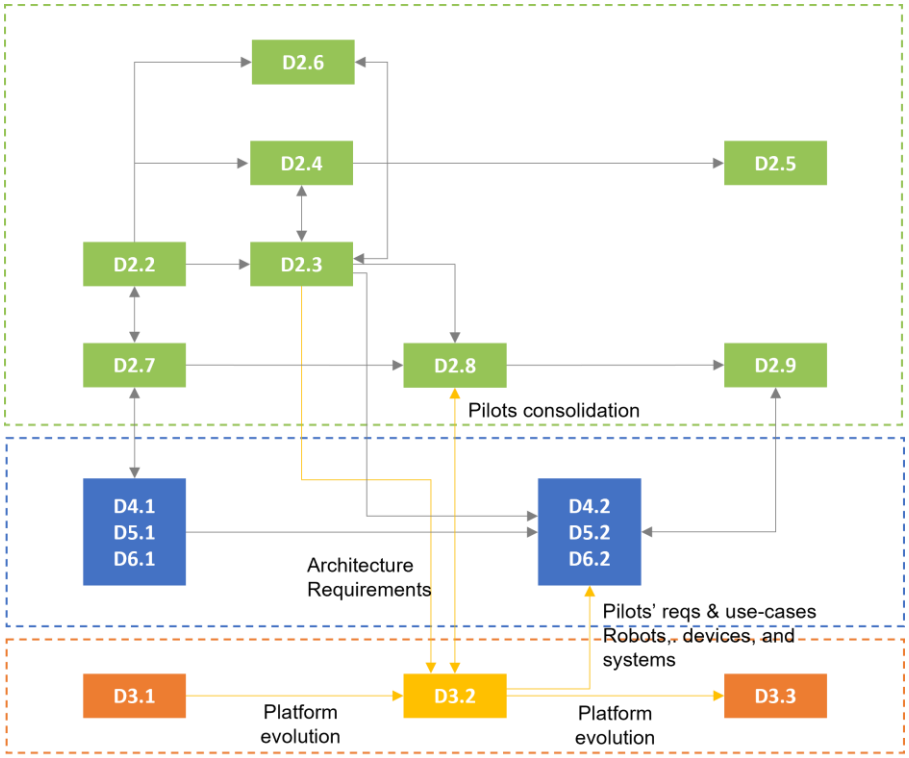

Figure 1 Deliverables linked to D3.2

The content of D3.2 summarises the implementation status at the end of the second year of the project for the prototypes built in these five tasks, considering as starting point the results reported in the previous document D3.1.

The corresponding functionalities have been implemented taking into account the requirements (functional and non-functional) that were specified in D2.2 and the final technical architecture proposed by D2.3. The evolution of the prototypes with respect to the previous version is also the result of the tests and validation activities done in with the three project pilots, which will be described in detail in D4.2, D5.2 and D6.2. A consolidated and common assessment is included in D2.8.

As introduced in D3.1, the five tasks of WP3 are following continuous, iterative and agile development methodology to develop the multiple components that are part of the FlexiGroBots platform. New versions of the platform are being released with more advanced functionalities so that they can be integrated, demonstrated and assessed by the pilots. The

final results will be reported on D4.3, D4.4, D5.3, D5.4, D6.3 and D6.4 at the end of the last year of the project. Again, a holistic vision will be offered through D2.9.

A final version of this deliverable, D3.3, will be released in M36 of the project, which corresponds to December 2023.

1.3 Structure of the document

This document maintains the format of the previous deliverable, as it is composed of the same seven sections:

- Section 1 includes the introduction to the document and the general content –such as document objectives, the relationship with other project activities, and clarifications addressing the outcome of the M18 review.
- Sections 2 to 6 correspond to the different tasks in WP3, specifically:
    - Section 2 updates the status of the *Artificial Intelligence platform* (T3.1), including the main advances in the development of pipelines.
    - Section 3 includes the status of project's *Common data enablers and services* (T3.2), focusing on the Data Space deployment and its integration with pilots.
    - Section 4 contains the latest advances in regards of the *Geospatial enablers and services* (T3.3).
    - Section 5 collects the details corresponding to the development of the *Common application services* (T3.4), including both those in which a better performance has been achieved and those that are still under development.
    - Section 6 compiles the progress regarding the architecture, communication standards and component development of the *Mission Control Centre* (T3.5).
- Section 7 includes the milestones reached during the last months of the project –which are collected in this document– and the relationship with the work for the coming months.

1.4 Addressing M18 review outcome

The consortium has decided to include specific content to address the main aspects that were highlighted in the outcome of project's M18 review and that need to be explicitly clarified. Given the standardized structure of the document, this specific supplemental subsection provides more flexibility to include explanations and maintains the ease of comparing with previous and future versions of the deliverable.

The content in this subsection has been arranged in four blocks corresponding to the most signalled aspects to be improved or clarified in the project. The first block of content corresponds to the relationship of the platform with its definition of architecture, its implementation, and the integration of components. Then, specific content regarding the

project's specialization in agriculture and how trustworthiness is being considered and addressed within the project is included. The last part of this subsection compiles platform's components and clarifies their origin. This is, which components have been developed prior to the project, which are external (third-party software used in the platform), and which are new or have been tailored to meet the specific requirements of the project.

1.4.1 Platform's architecture, implementation, and components integration

The FlexiGroBots' platform architecture, as described in D2.3 –which outlines the final requirements and specifications of the platform–, serves as a reference for the implementation of the respective WP3 building blocks and provides guidance for the integration, execution, and validation activities of the three pilots in WP4, WP5, and WP6 (refer to D2.3 for a complete specification including functional and non-functional requirements, and use cases). The aim of this subsection is to include an explicit link between the defined architecture in WP2 and the work done corresponding to each task in WP3. For this, the architecture diagram corresponding to the platform provided in D2.3 is also reused here (Figure 2), on which labels have been added specifying which task of package 3 corresponds to the implementation of the different components defined in WP2:

- T3.1 encompasses *"a series of modules that offer an integrated solution, leveraging Machine Learning and Deep Learning techniques through Machine Learning Operations (MLOps) and Automatic Machine Learning (AutoML) paradigms, to extract value from the information collected from various devices, robots, and other components".*
- T3.2 includes the corresponding tasks related to the "*information management and exchange between multiple stakeholders in a secure and sovereign manner, in accordance with the principles of the Data Spaces concept and the International Data Spaces Association (IDSA). It also provides access to real-time or near real-time information and historical data".*
- T3.3 incorporates the components to "*processes drone and satellite remote sensing products, exposing the results through standard Open Geospatial Consortium (OGC) interfaces".*
- T3.4 comprehends *"off the shelf" services for general precision agriculture through an "AI as a Service (AIaaS)" approach"*.
- T3.5 involves the development of the *"components to provide the solution with the capability to plan and execute complex missions that involve fleets of heterogeneous robots through a powerful Mission Control Centre (MCC)"*.

Platform's components related to pilots (WP4-6) are also depicted in Figure 2.

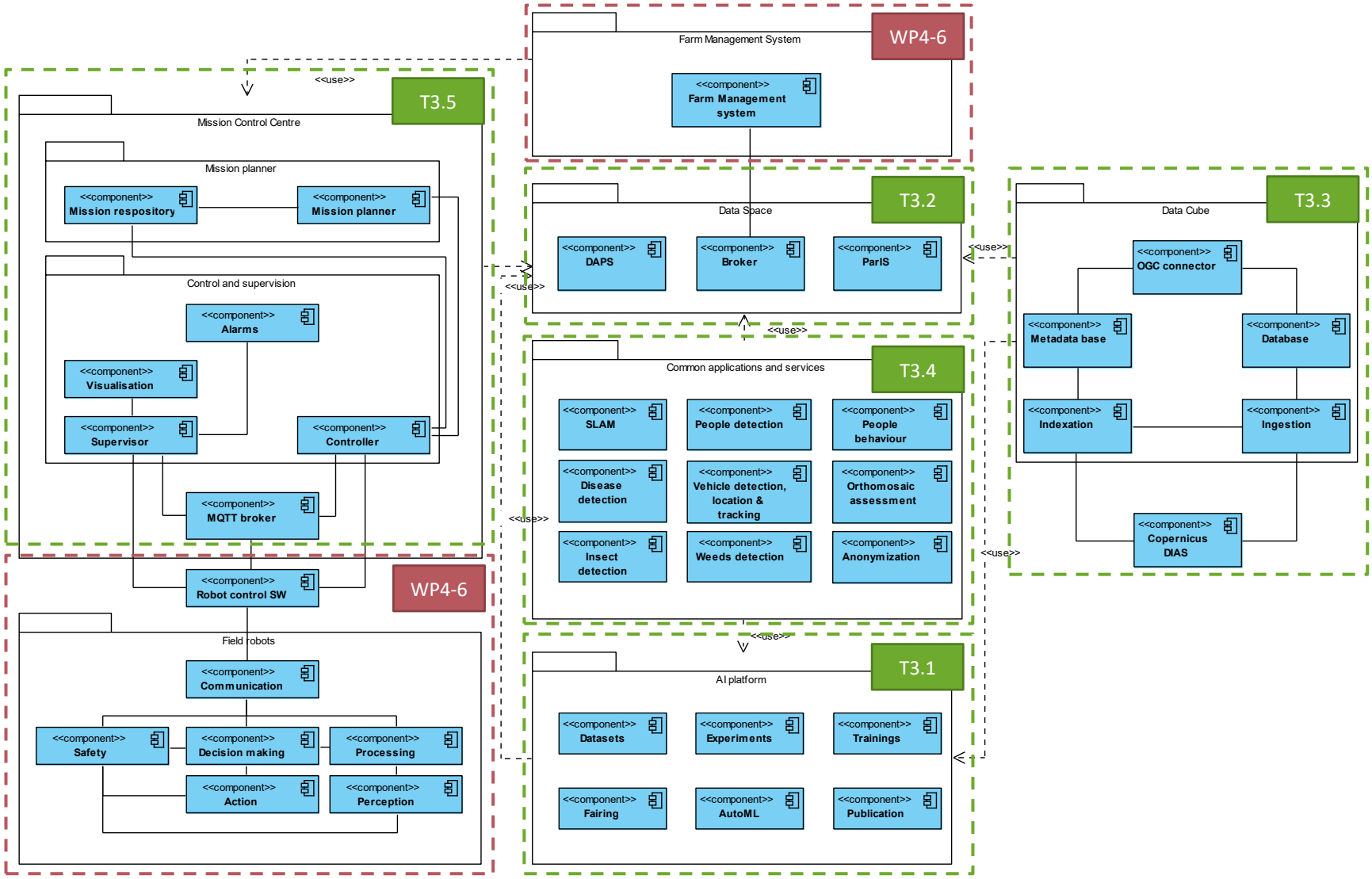

Figure 2 FlexiGroBots platform development depicted in D2.3, labelling the scope of the tasks in WP3

1.4.1.1 Platform components integration

The consortium recognizes the importance of clarifying the integration status of the components that enlist the integrated platform. At this stage of the project (M24), as the development of the platform continues to progress, more information in this regard can be provided. This will not only help in overcoming the information gap indicated in the M18 Review, but also in making it easier to populate the platform with solutions developed from the pilots, ensuring the smooth operation of the solution. In light of this, the effort of the last year of the project will be put on the integration and standardization of these crucial elements and their relationship to support the FlexiGroBots' platform as a whole, with particular focus on the Data Space as its backbone and high-level standards for the AI assets.

For the time being, section 3 includes a detailed explanation of the architecture defined to implement the integration of two important components of the platform, namely the AI subsystem and the Data Space. This combination will allow users of the AI subsystem (Kubeflow) to access the data catalogues provided in the Data Space (which could be used for model training, testing, etc.) in a unified, integrated and, secure way. The integration of the geospatial enablers and services and the Data Space is, at this stage of the project, still to be defined given the lack of compatibility between the components' interfaces. More information on this integration will be reported later in the project. The integration of the MCC with the Data Space is something that has already been worked on in recent months in line with the definition of the platform architecture. More information can be found in sections 3 and 6.

In addition to this, information regarding the integration of the common application services is included in the dedicated section 1.4.1.2 of this document. Moreover, further details regarding the integration with the AI marketplace (AI4EU) [2] and the AI-on-Demand (AIoD) platform [3] are included in section 2.3.

1.4.1.2 AI features integration

The deployment of common applications as inference services in a distributed manner is subject to the availability of HW resources in the platform's Kubernetes cluster (see sections 2.2.1 and 2.6 for more information) and the resources available at the edge. At this intermediate point in the project, work still needs to be done to finalise the definition of how the inference services will be distributed and hosted.

The current state-of-the-art computer vision models demand a large amount of computational resources, especially GPUs, which are indispensable to be able to work with execution times that can be considered as real-time. Currently, there is only one GPU (model Nvidia Tesla T4) in the AI platform server, in addition to the computational capacity of the CPU. This HW is absolutely insufficient to simultaneously support several inference services.

Therefore, there are two possible alternatives: introduce many more HW resources to the cluster, which will most likely lead to exceeding the budget allocated to this type of resources in the project; or use the resources available in the cluster to respond to offline requests, that is, those that do not require an instant response, and study the use of hardware resources on the edge, such as laptops or other devices with high computing capacity, to process the images and videos of real-time applications. Both alternatives would rely on using a messaging broker, such as MQTT [4], to add an interface layer that allows launching requests and receiving responses in the same way regardless of the computing device where the inference service is deployed. This is still under testing and more details will be provided later in the project.

1.4.2 Specialization on agriculture

Regarding the lack of specialization of common applications towards agriculture pointed out by the reviewers in the M18 Review, actions are being taken to reverse this impression, and top priority will be given to the development of more specific applications.

Several applications were labelled as too generic, of non-specific use in the context of agriculture, such as image anonymization, estimation of the position and distance of people close to UGVs, automatic generation of datasets, or detection of human actions. It is understandable to question the apparent priority given to these more generic applications, since they are the most advanced at M18.

The main reason for this greater progress is that to a large extent these applications have been able to be implemented without the need for components resulting from the work carried out by the pilots, with the exception of small data sets to validate the algorithms. On the contrary, many of the specific applications require the integration of the models and pipelines developed in the context of the pilots, which has meant having to wait for the second half of the project to give margin to obtain results and to be able to proceed to focus on them.

In any case, all the applications, both those already fully developed and those that have yet to evolve, meet a real need drawn from the project requirements, and provide added value in the context of agriculture. In the final paragraph of the description of the status of each of the tools –section 5 of this document– the resulting added value is reflected in each case.

On the other hand, it is considered that in the same way that negative conclusions can be drawn about the lack of specialization of some of the applications, it is also possible to draw value from the fact that an application developed in the project can be extrapolated to other industries, and that therefore what is really interesting is to cover both the more generic and the more specific aspects. In response to this comment, the efforts available during the remaining months of the project will be focused on enhancing the specific functionalities planned for agriculture, such as the detection of insect pests in crops, weeds, or fruit diseases.

1.4.3 The AI Platform and Trustworthy AI

The definition of Trustworthiness used in FlexiGroBots follows the definition proposed by the EU High-Level Expert Group on AI (AI-HLEG) [5]. This means, that an AI is considered to be trustworthy when it is lawful, ethical, and robust. The AI-HLEG operationalises these three general principles in seven requirements regarding *"(1) human agency and oversight, (2) technical robustness and safety, (3) privacy and data governance (4) transparency, (5) diversity, non-discrimination and fairness, (6) environmental and societal well-being and (7) accountability*". These requirements can be assessed with the Assessment List for Trustworthy AI (ALTAI) [6]. The FlexiGroBots project uses this corresponding ALTAI questionnaire as the basis for the internal assessment of trustworthy AI. Deliverable D2.6 provides comprehensive details on this internal assessment. This short section summarises the main components of this assessment that are linked specifically to the FlexiGroBots AI Platform (WP3).

The trustworthy AI assessment in FlexiGroBots follows several steps: First, an initial round of interviews with the platform developers (and each pilot) was conducted in August and September 2021 based on the ALTAI. As it is stated in D2.6, *"the interviews were led by CEPS with technical and well as managerial members of the respective partners. This setup followed the recommendation that the ALTAI "is best completed involving a multidisciplinary team of people."* [6]. *During these interviews, the group discussed and filled in each question of the ALTAI questionnaire. Based on the discussion and findings during the interviews, CEPS drafted a list of initial recommendations tailored to the platform (and each pilot). These recommendations were shared with all partners in November 2021 and partners had the opportunity to provide feedback. All partners involved decided to follow-up on specific recommendations and questions which came up during the initial assessment in more targeted follow-up interviews. The first follow-up interviews focused on questions related to data protection and privacy"*. These interviews will continue in the second reporting period, for example with targeted interviews on safety-related questions. All resulting recommendations from the first half of the project are available in D2.6.

In order to link the FlexiGroBots assessment of trustworthy AI to the AI-HLEG definition of AI, each recommendation was specifically mapped to at least one of the seven requirements for trustworthy AI. The full list of recommendations specifically for the platform are available in D2.6, chapter 5.2, table 6. Moreover, as a means of harmonising the FlexiGroBots approach to trustworthy AI, a similar structure was applied to the assessment of each pilot.

Furthermore, regarding the aspect of lawfulness, the consortium partners have reviewed regulatory frameworks and private standards that are relevant to the project. Details on this review, e.g. on legal standard related to data protection, autonomous vehicles or AI, are available in D2.6, chapter 4. These legal reviews continue during the second half of the project and will focus on aspects such as the New Machinery Regulation, the AI Act, intellectual property, and contracting standards. Moreover, a targeted interview with the platform

developers was organised by CEPS, to discuss questions related to data protection for the platform. Lastly, the state of the art has been advanced in the assessment and monitoring of the Trustworthiness of AI, through the expansion of model cards for model reporting and data sheets for reporting data sets. The project builds upon the model cards standard from Mitchell et al. [7] and the datasheets standard from Gebru et al. [8] and adapts them to the specific context of autonomous robots and agriculture. The project created concrete templates for model cards and data sheets (see chapter 5.6 and 5.7 in D2.6), which will be filled in for each dataset and AI model developed in the project. These templates provide a concrete tool for standardising the reporting and monitoring of trustworthy AI standards in the project. Model cards and dataset datasheets corresponding to WP3 are included in Annex A.

Furthermore, building upon these standards, the project has also started discussions with the AI4Europe platform for integrating aspects of these standards into the AI4Europe AI asset upload interface.

1.4.4 Platform components inventory

This last section of the document introduction includes a list of the components that make up the FlexiGroBots solution for the centralized platform. Only a list of the actual components is compiled here (see Table 1), so that it is possible to see at a glance what other modules or elements they are based on, and a very brief description of the work done in the project. Detailed information on each of these components –including a description of the added value by the project– can be found in the corresponding sections of the deliverable. So, refer to these for a detailed explanation.

Component	Base element	Purpose	Type	Work done within the project
AI Platform	Kubeflow	ML workflows implementation and inference services	Third-party (OS)	Deployment, configuration, and integration
	MinIO	Data Lake and storage		
Data Space (DS)	IDS-TestBed	Deploy the Data Space	Third-party (OS)	Deployment (adaptation to Kubernetes), configuration, and integration
Integration module (DS + AI platform)	Fast-API, Redis Database	Integrate the AI subsystem using the DS	Own	Development

Component	Base element	Purpose	Type	Work done within the project
Data Cube	Open Data Cube (ODC)	Facilitate the management and access (via OGC APIs) to hundreds/thousands of satellite and UAV imagery scenes	Third-party (OS)	Deployment, configuration, and integration
stac-generator.py	gdal-bin, python-numpy, boto3, shapely, rasterio, rio-stac, pystac	Python script for automatically generating the STAC metadata necessary for indexing and making aware the ODC of the new UAV datasets as they are available	Own	Development from scratch
Mapping Service	MapServer	Offer via standard OGC APIs access to the botrytis data (in vector-based or map format) extracted from the collected datasets in the Spanish pilot	Third-party (OS)	Deployment, configuration, and integration
	Postgres+Postgis	Geospatial database used for the storage of the botrytis data	Third-party (OS)	
SLAM	Zed2i stereocam	3D positional tracking and mapping	Third-party (OS)	Integration and testing of commercial device/SW
People Detection Localization and Tracking	Detic	Object/people detection	Third-party (OS) with own modules	Pipeline development (component integration)
	Clip + deepSort	Zero-shot multi-object tracker		
	DPT	Depth estimation		
People Action Recognition	mmaction2	Action sequence recognition	Third-party (OS)	Component integration and testing

Component	Base element	Purpose	Type	Work done within the project
Moving Objects Detection	TPH-Yolov5	Tractor detection	Own	Model training & pipeline development
	Clip + deepSort	Zero-shot multi-object tracker	Third-party (OS)	Pipeline development (component integration)
Orthomosaic Assessment Tool	OpenDroneMap	Orthomosaics + DTM + DSM	Third-party (OS) with own modules	Pipeline development (component integration)
Anonymization Tool	DeepPrivacy	Face detection + face generation GAN	Third-party (OS)	Component integration and testing
Automatic Dataset Generator Tool	clipRetrival + img2dataset	Raw image data collector	Third-party (OS) with own modules	Pipeline development (component integration)
	Detic	Labelling zero-shot		
	fiftyone	Dataset visualization		
Fruit Disease Detection	Detic	Fruit detection and segmentation	Third-party (OS) with own modules	Pipeline development (component integration)
Pest Detection	Detic	Insects and traps detection and segmentation	Third-party (OS) with own modules	Pipeline development (component integration)
Weed Detection	Detic	Weeds detection and segmentation	Third-party (OS) with own modules	Pipeline development (component integration)
MCC	Mission workflow planner	Mission workflow planning	Own	SW specification and development
	Mission workflow controller	Mission workflow control	Own	SW specification

Component	Base element	Purpose	Type	Work done within the project
				and development
	Mission reporter	Mission report creation	Own	SW specification and development
	Mission repository	Storage for mission descriptions	Third-party (OS)	Use of existing file system
	Mission management GUI	User interface to mission workflow management	Own	Development
	Data space interface	Interface to Agriculture data space	Third-party (OS) with own extensions	IDSA connector extended
	Fleet controller	Sending commands to robots in the fleet	Third-party (OS) with own extensions	Extension to QGC software
	Fleet supervisor	Visualisation of robots' location and status at the field.	Third-party (OS) with own extensions	Extended use of QGC
	MQTT broker	Transfer of MQTT messages between robots and fleet controller and supervisor	Third-party (OS)	Use of OS broker
	MCC communication	MCC interfaces to robots and other components	Own	Development
	Robot task planner	Robot route planning	Own	Development

Table 1 Summary of the building blocks that compose the FlexiGroBots platform

2 Artificial Intelligence platform

The Artificial Intelligence (AI) platform is the core element of the FlexiGroBots platform that facilitates image processing and recognition, through different deep learning (DL) and machine learning (ML) models. Its AutoML functionality allows users to achieve high performance of image segmentation and classification.

2.1 Implemented functionalities

During the previous reporting period, we advanced from having implemented individual components of the platform, to having the final prototype with fully functioning elements and communications between them. The results of the project in terms of data and models will be shared through the **Agricultural Data Spaces (ADS)** and the AI on-demand platform resulting from the **AI4EU** project. The main results achieved in the previous reporting period are the following:

- Having allocated appropriate storage and computing resources, the user can follow two pipelines shown in Figure 3. The **training pipeline** (green box) is intended for building optimal deep learning (DL) models based on the training data and labelled images, while the **test pipeline** (red box) is intended for executing DL on input images during regular operation of the platform within pilot use-cases.

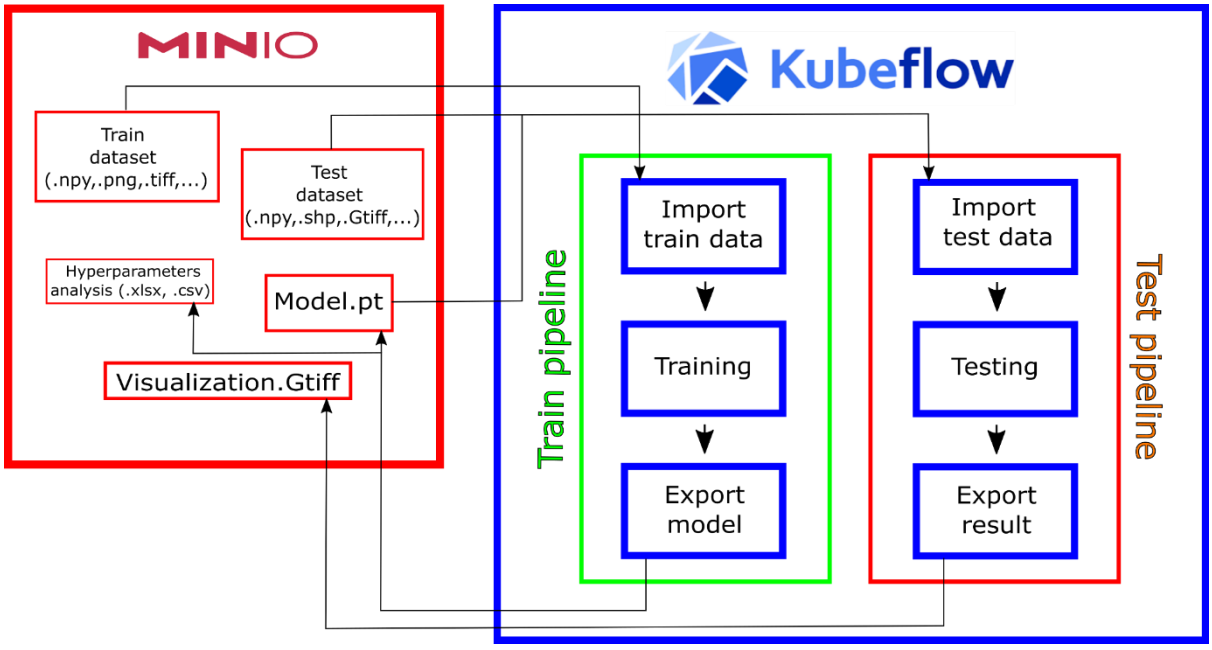

Figure 3 AutoML procedure with training and testing pipelines

- The **toolbox** has been enriched with additional libraries and packages so that it now includes deep learning libraries (PyTorch being the primary one), standard Python

libraries (numpy, pandas), machine learning pipelines (scikit-learn), image processing libraries (scikit-image, OpenCV) and processing of geospatial data (gdal)

- The drafted **workflow** has been implemented and now encompasses the whole life-cycle of ML models. MinIO is used for import/export of data/models, standard image processing techniques are used for pre- and post-processing, different DL architectures can be explored along with a number of different hyperparameters and this process is governed by the AutoML functionality.
- **MinIO** was implemented as a highly optimised data storage that facilitates integration of data from all Pilots. An HTTP connector has been implemented within MinIO as an edge gateway for seamless communication with Pilots' Farm Management Systems (FMS), and more generally with ADS. MinIO, however, can also accept data from other sources, such as user's storage or cloud drive, using HTTP or other protocols.
- The **AutoML** functionality covers different neural network architectures (UNet, UNet++ etc.) and explores different hyperparameters (model, learning rate (LR), LR Scheduler, Lambda, step, batch size, number of epochs). Model performance is evaluated based on introduced metrics which depend on the tasks (e.g., Intersection over Union (IoU)). The result of the hyperparameter optimisation is the best-performing model itself, coupled with an *xml* file with the list of final properties of the model (Figure 4).

	Model	Learning rate (LR)	LR Scheduler	Lambda	Step	Batch size	Number of epochs	IoU Background	IoU Blueberry
1	Unet++	0.01000	multiplicative	1	5	20	20	0.8	0.7
2	Unet++	0.00100	multiplicative	1	10	40	20	0.5	0.3
3	Unet++	0.00100	multiplicative	2	5	20	20	0.5	0.6
4	Unet++	0.00010	multiplicative	2	10	40	20	0.3	0.4
5	Unet++	0.00010	multiplicative	1	5	20	20	0.4	0.7
6	Unet++	0.00010	multiplicative	1	10	40	40	0.7	0.5
7	Unet++	0.00001	multiplicative	2	5	20	40	0.4	0.6
8	Unet++	0.00001	multiplicative	2	10	40	40	0.5	0.7
9	Unet++	0.00001	multiplicative	1	5	20	40	0.3	0.4
10	Unet++	0.00001	multiplicative	1	10	40	40	0.2	0.5

Figure 4 After an experiment is finished, results are stored inside an XML file with hyperparameters and results for row detection

- A dedicated **GitHub repository** was set up for the trained AI models for easy sharing of implemented models between the pilots and use-cases. We developed a **generic Docker container** for these models and an example can be found on the link in Section 2.7. It offers multi-platform support (x86, ARM, GPU, TPU) for the specific hardware for easy scalability over different platforms and use-cases.
- **Access** to the AI platform has been made available through: Jupyter Notebooks, Software Development Kits (SDKs) and/or Command Line Interfaces (CLI), while the results are accessible through **TensorBoard** integrated in the Kubeflow [9] (Figure 5).

This feature provides detailed information about the training loss, performance metrics for different configurations of hyperparameters through the epochs. It can be also very useful for model/training analysis, selection of the best performing models, and for redefining a new range of values for hyperparameters.

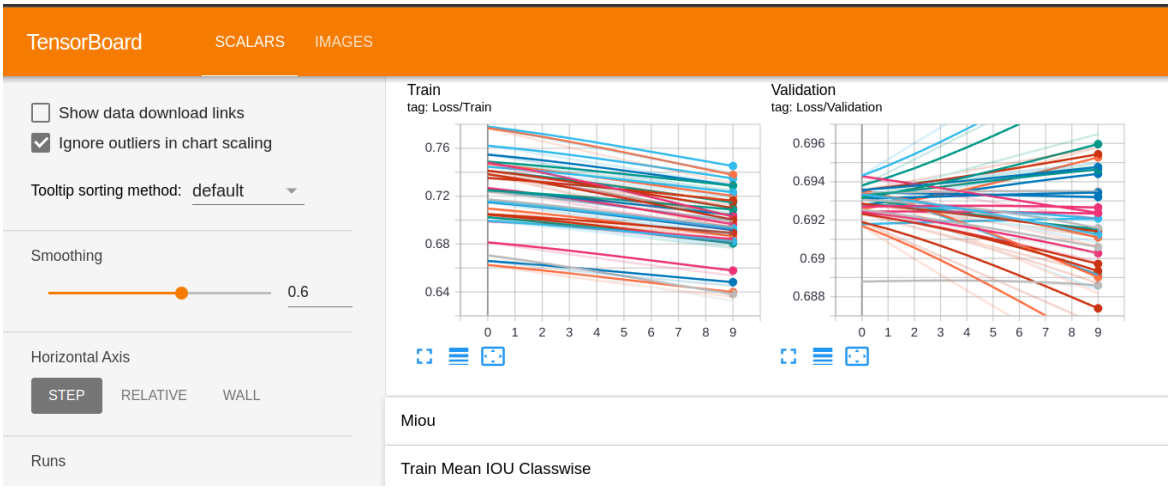

Figure 5 Training results are accessible by TensorBoard

MinIO, KubeFlow, TensorBoard and similar tools used in the project are developed by different software companies and communities. The main contribution of FlexiGroBots is in the domain of system integration. The server has been set up from scratch and the components installed, connected, and streamlined to cover the whole ML lifecycle from data to actionable insights. Regarding the models on the platform, we used state-of-the-art models and fine-tuned them to fit the input format of UAV images and the output format for the subsequent UGV operations.

In addition, the AI platform will be connected with other platform's components through the Data Space. This connection will be explained in more detail in section 3 (Common data services), where the modules linking to the AI platform will be described.

2.2 Requirements

2.2.1 Technical requirements

2.2.1.1 MinIO

MinIO can be defined as an open-source storage server, this solution has been deployed using a Kubernetes cluster (v1.21.9) on a virtual machine with the next hardware characteristics:

- Intel Xeon Processor (CascadeLake) 6 CPUs
- 502 GB storage
- 12 GB RAM memory

2.2.1.2 Kubeflow

As stated in D3.1, Kubeflow [9] relies on Kubernetes for its deployment. The cluster in use has been extended to meet the computing needs of the project and also to include a GPU. It currently consists of the following nodes:

- Master:
  - CPU: 12 cores
  - RAM: 32GB
  - Disk: 4TB
  - Operating system: Ubuntu 20.04.2 LTS
- Worker:
  - CPU: 64 cores
  - RAM: 124GB
  - Disk: 890GB
  - GPU: Tesla T4
  - Operating system: Ubuntu 20.04 LTS

2.2.2 Functional requirements

Not relevant.

2.3 Data models

The AI platform has been developed to be agnostic towards the type of the data, as different pilots will have different UAV/UGV images (RGB, multispectral, hyperspectral…) and possibly additional data (e.g., shapefiles). Data is fetched from external sources (ADS, FMS, user storage…) through HTTP and placed on MinIO, from where it continues its path to Kubeflow.

The integration of the AI4EU marketplace [2] and the AI-on-Demand platform (AIoD) [3] is a mandatory aspect of the project that still requires implementation. Despite this, additional information beyond what is included in D3.1 can now be provided. These platforms are interconnected, enabling users to upload a diverse range of AI assets, such as datasets, ML models, Jupyter Notebooks, repositories, and Docker containers. In detail, the AI4EU marketplace requires Docker containers to include Protobuf[1] [10] format for use on the platform. The adoption of this format in the AI subsystem will require specific modifications and adaptations of the containers in use because Kubeflow does not use this standard for the

[1] Protobuf , short for Protocol Buffers, is a language-agnostic data serialization format developed by Google that permits defining a data schema in a simple language and utilizing that schema to serialize and deserialize structured data in a compact binary format. It is frequently utilized for inter-process communication, data storage, and network communication, particularly in high-performance applications where efficiency and data size are critical.

underlying containers supporting pipeline components or inference services. As a result, the consortium must prudently assess the investment required for this adaptation. In contrast, the AIoD platform is more flexible, providing users with greater freedom. It permits uploading almost any type of AI asset in any format, creating a more open and adaptable environment for collaboration, and sharing.

As a preliminary plan, FlexiGroBots consortium intends to integrate its assets in the AIoD marketplace. This integration path, though requiring further clarification, will involve submitting AI assets that comprise the pilot applications, considering any interoperability or compatibility components that the platform imposes. The most established algorithms for image detection (agricultural vehicles from UAV footage, blueberries, and grapes) and the orthomosaic tool will be the first ones to be uploaded to the platform.

2.4 Application Programming Interfaces (APIs)

As stated in D3.1, *"Kubeflow provides a Python SDK to manage and execute ML pipelines. It also provides a RESTful API and supports main ML libraries like TensorFlow, PyTorch or scikit-learn".* There are no updates in this regard.

MinIO [11] offers an API that can be accessed through the use of different programming languages such as Python, JavaScript, and C#, allowing interaction with the data lake in a more versatile way using scripts. For this project, a small library has been designed using Python scripts that allow uploading, downloading and representation of buckets [12]. With this library, it is managed MinIO from creating buckets to uploading and downloading data. In addition, it is possible to set up the administration of the data lake with different sorts of permissions. All MinIO instructions are on the official web page of the data lake.

2.5 Graphical User Interfaces (GUIs)

The ML pipeline can be executed through Kubeflow's GUI. Screenshots in Figure 6 and Figure 7 show how experiment runs are initiated and how the boundaries of hyperparameters are defined.

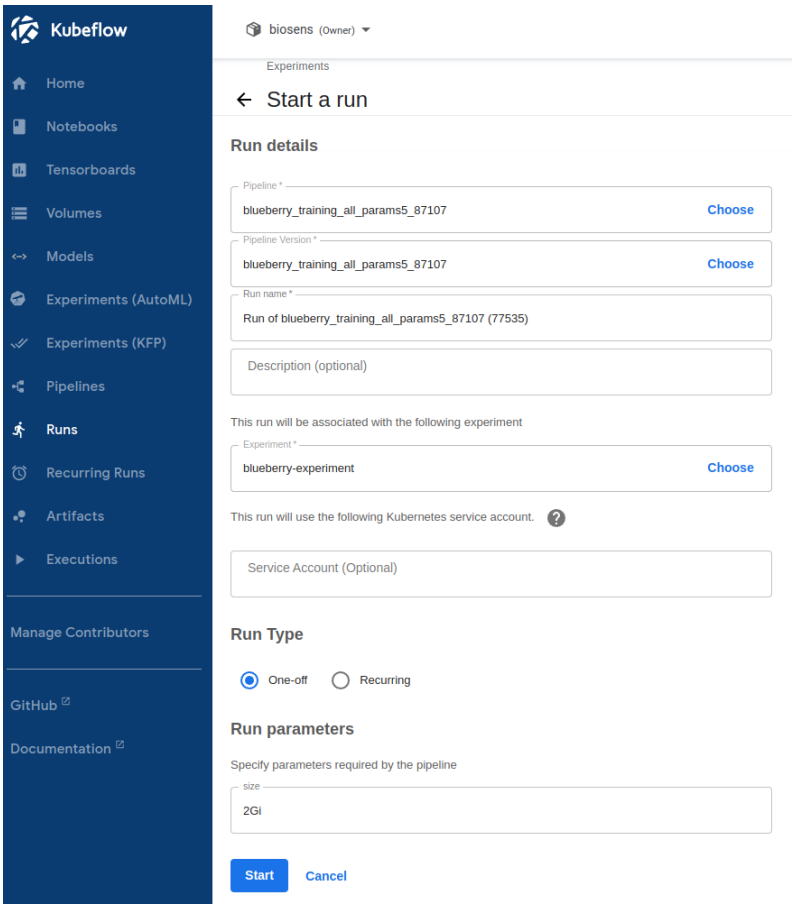

Figure 6 Initiating an experiment run through Kubeflow GUI

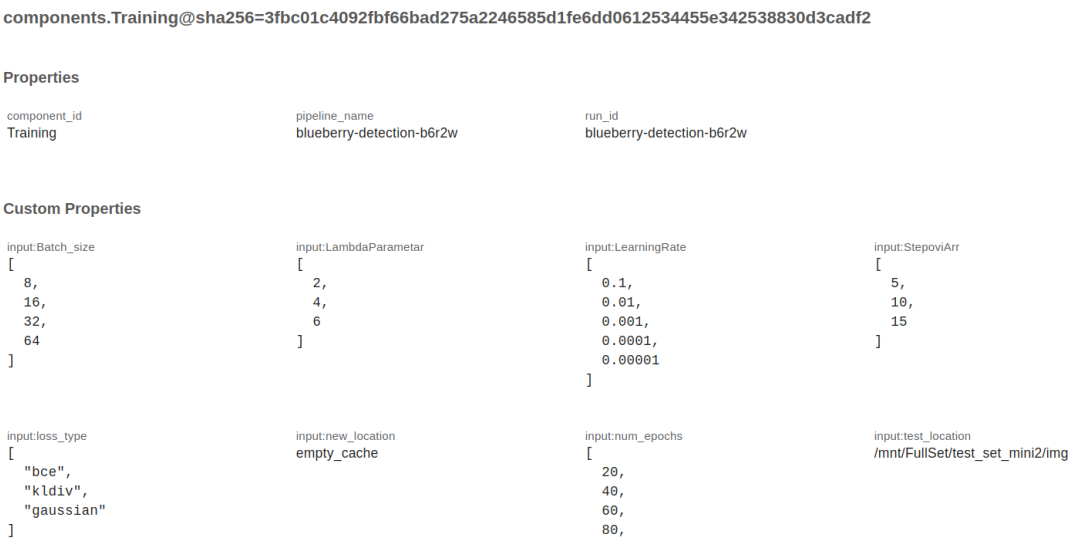

Figure 7 Definition of hyperparameters within the training pipeline through Kubeflow GUI

Otherwise, MinIO offers a GUI that allows working with the data lake in a more visual and simple way. This GUI is prepared to work using the browser, in addition there is a manage system used to difference the permissions of each user. Figure 8 shows the login web page to MinIO.

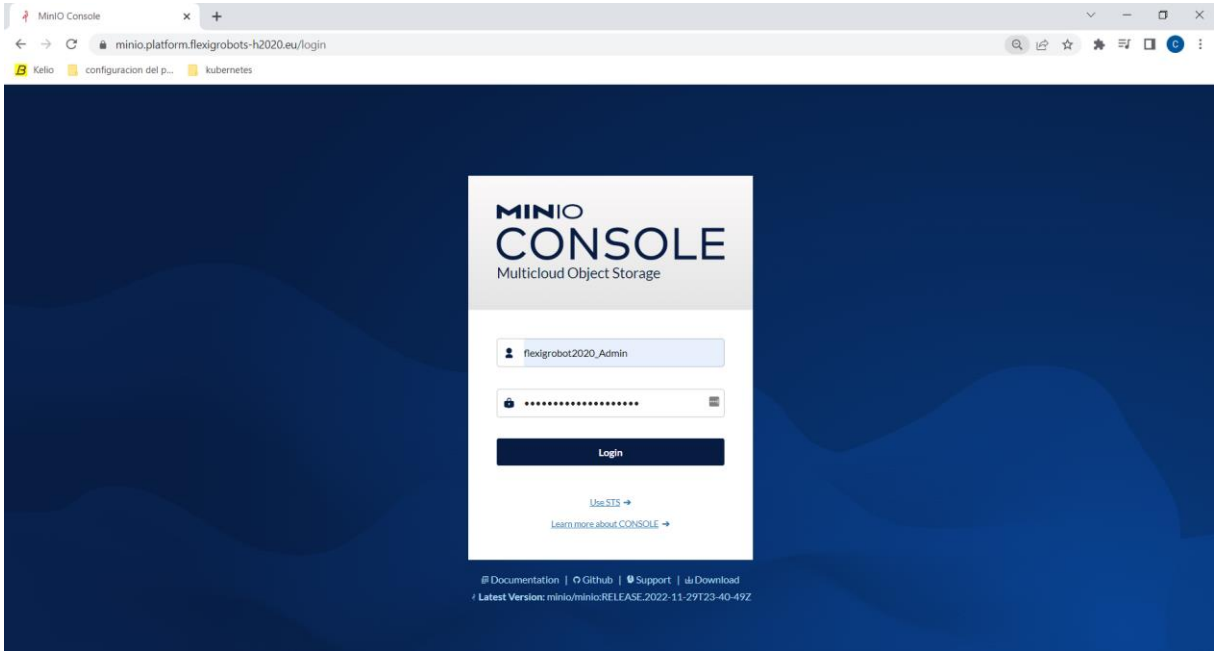

Figure 8 MinIO's log in web interface

Figure 9 shows the aspect of the data lake, from this GUI it is possible to configure the user permission, the buckets, notifications, lambda functions and monitoring.

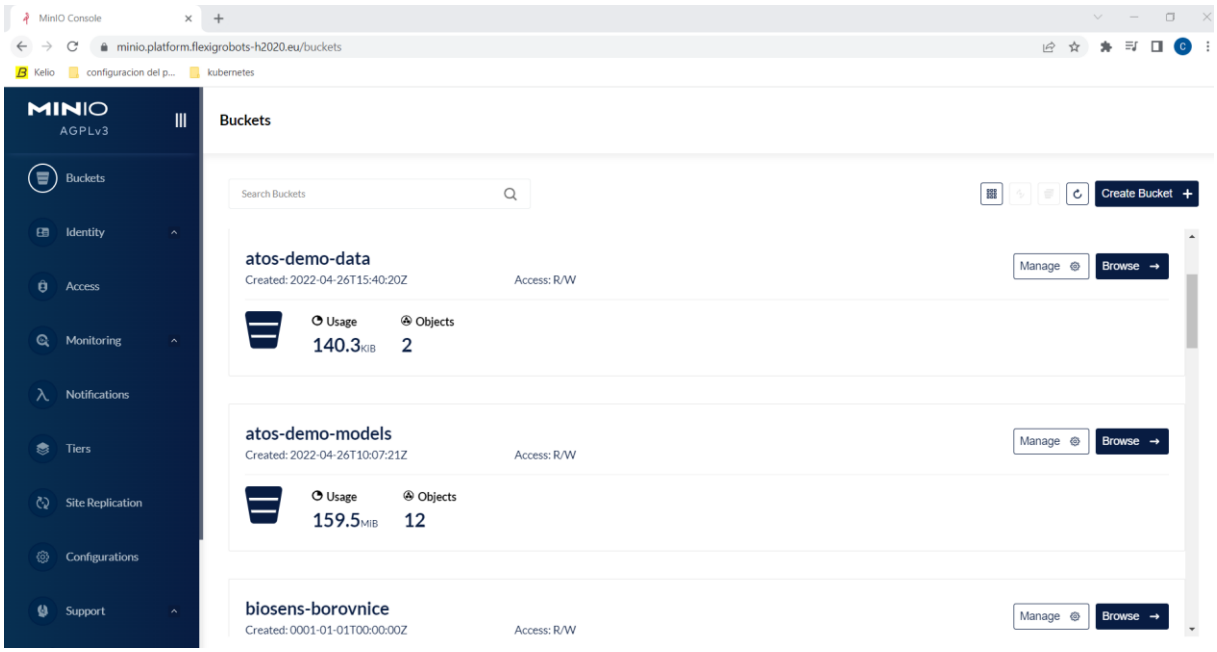

Figure 9 MinIO main interface

2.6 Installation

The installation of Kubeflow was done on ATOS's Kubernetes cluster based on the available Kubernetes manifests[2] available for bare-metal deployments. The additional steps required to achieve a fully working deployment are included in project's repository[3]. Among the milestones achieved, the following stand out:

- Enabling of a centralized authentication system based on Dex[4] and connected to FlexiGroBots *organization* in GitHub. The users belonging to a specific team within the organization will have access to the Kubeflow instance.
- Deployment of Nvidia's GPU Operator[5] to manage workloads making use of GPU in Kubernetes. Additionally, *time-slicing* configuration for oversubscribing GPUs in Kubernetes[6] was applied. This enables multiple workloads –which might belong to one or multiple users– to use the same GPU concurrently (GPU oversubscribing is disabled by default in Kubernetes).

We also installed a wide variety of Python libraries for image processing and deep/machine learning (PyTorch, scikit-learn, scikit-image…), as well as auxiliary libraries for data handling. The server was configured to support model training/testing/execution on GPU servers.

MinIO offers several ways to install it, from dockers and Linux to Kubernetes. For FlexiGroBots, Kubernetes has been the established way of MinIO; this is due to the kind of project architecture. The easy option to install MinIO is launching a set of manifests. These manifests are public in the project repository [13]. In addition, an URL has been associated with the data lake to make it public to the project's users. Finally, it has been configured the Kubernetes cluster to access MinIO from any browser using a public URL: https://minio.flexigrobots-h2020.eu

2.7 Prototype availability within FlexiGroBots

- MinIO is available at https://minio.flexigrobots-h2020.eu
- Kubeflow is available in https://kubeflow.flexigrobots-h2020
- As an example, a Pilot 3 AI model developed through the AI platform was wrapped using the Generic Container and is available at: https://github.com/FlexiGroBots-H2020/AI-platform/tree/master/kubeflow/blueberry_row_detection

[2] https://github.com/kubeflow/manifests
[3] https://github.com/FlexiGroBots-H2020/AI-platform/blob/master/kubeflow/README.md
[4] https://dexidp.io/
[5] https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/overview.html
[6] https://developer.nvidia.com/blog/improving-gpu-utilization-in-kubernetes/

2.8 Release planning

As of 12/2022 we can confirm that the final prototype of the platform has been developed. AutoML, which is the core functionality of the system, has been implemented and tested. However, there are some ongoing tasks, in which there was a slight lag in the development, due to the fact that the architecture based on Kubeflow and MinIO proved to be less stable than initially envisaged, and which took additional efforts for setting up. However, this lag will not affect project implementation as the ongoing tasks will be finished during the first quarter of 2023, well before the next blueberry/rapeseed/vine growing season.

- 01/2022:
  - Integration of AI platform into the ADS.
  - Integration of AI platform with systems from Pilots 2 and 3.
- 02/2022:
  - Interoperability with AI4EU.
  - Integration of the AI Platform with pilots' systems.
- 03/2022:
  - Additional features, feedback from pilots.
  - Final version of the platform

User story ID	User story	Priority	Story points
AI_US_01	Development of an IDSA connector for MinIO	High	3
AI_US_02	Integration of Kubeflow with AI4EU for re-using and publication of artefacts	High	5
AI_US_03	Testing of AutoML operators	High	1
AI_US_04	Integration of models for common application services	Medium	7
AI_US_05	Integration of models for pilot 1	High	8
AI_US_06	Integration of models for pilot 2	High	8
AI_US_07	Integration of models for pilot 3	High	8
AI_US_08	Optimisation for multiple architectures including ARM processors	Medium	13
AI_US_09	Integration of codecarbon library	Low	2
AI_US_10	Performance monitoring and retraining functionality	Medium	21

Table 2 User stories for the AI platform

3 Common data services

As mentioned in the previous D3.1 "Common data services" [14], this work package aims to guarantee equal opportunities and trust in data sharing. Thence, Atos has deployed a IDSA data space. The main goals of IDSA are to preserve security and data sovereignty in the data exchanged; for this reason, Atos has designed all needed manifests to deploy an IDSA DS like a production-ready solution.

IDSA published a repository [15] with a real TestBed; this is an example of a real DS. The design of this TestBed is oriented to use in local/development environments; for this reason, and the project requirements, it has been compulsory to adapt the TestBed from a local environment to a production environment.

Migration has been switched from the Docker Compose technology to a technology-oriented cluster, Kubernetes [16]. The aim is to provide an Open Source and production-ready solution to work in a real environment.

Kubernetes uses a set of files named manifests that define the behaviour of each component. For this reason, in this second part of the project, all manifests have been developed to deploy the system in a production Kubernetes cluster. In addition, these files have been configured to work in a cloud cluster (Rancher cluster) and allowing to access the DS from the Internet.

In addition, another repository [17] has been created within the FlexiGroBots organization where the steps for deploying this system are detailed. Due to the IDSA policy, as explained in section IDS-G from IDSA official repository [18] where the steps for the deployment of this system are detailed, "*IDS-G is the point of truth for specifications of the IDS and its components in the IDS GitHub page. It is also public for everyone and contains the approved specifications that were confirmed by the IDS Technical Steering Committee (IDS-TSC) and the IDSA Working Groups. IDS-G publishes quarterly releases with new approvals by the Working Groups and the TSC*".

As mentioned in the previous section, an integration architecture has been designed to join the AI platform with the rest of the parts of the project. This will allow authorised users to use the AI platform and request data from the consortium companies to train their models. This integration will be managed through the Data Space, using the DS connectors as a gate to access the private dataset.

The last point will be the Data Space integration with the Mission Control Centre. Currently, the Mission Control Centre uses a Data Space deployed in a local machine to perform tests and work in parallel with the rest of the consortium parts. But for the last stage of the project, the integration of both parts will be a high priority.

3.1 Implemented functionalities

The implemented functionalities are summarized below:

- In order to evaluate that the system works correctly, it has been evaluated that the functionalities of the DS do not change independently of the technology used, being able to affirm that with Kubernetes, it works in the same way as with Docker compose but in a production environment. Figure 10 shows this migration and how the architecture has been adapted to the project. The left block is the IDS-TestBed deployed using Docker-Compose; this deployment runs in a local machine environment; because of this, external connectors cannot access to the DS. On the other hand, the right block represents the same IDS-TestBed but running Kubernetes technology. The external elements can connect to the DS and exchange data in this case.

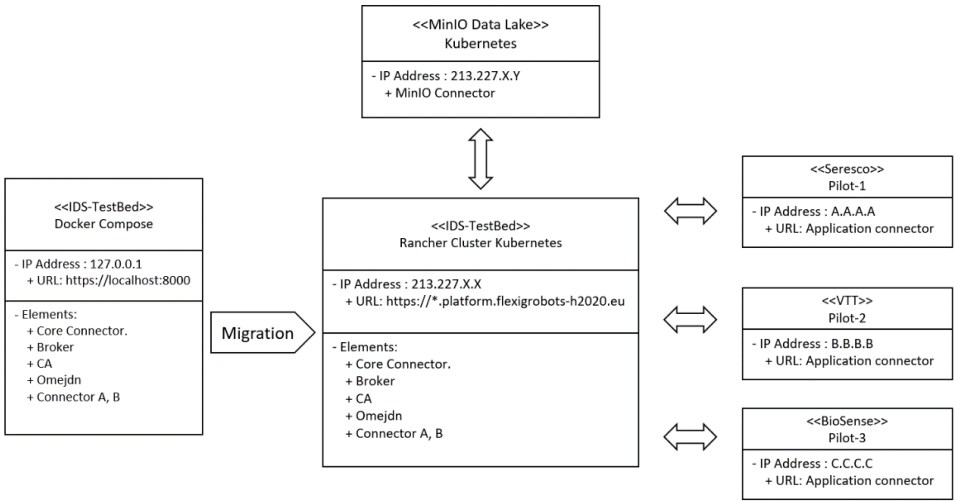

Figure 10 Migration from Docker Compose to Kubernetes

- The DS cluster is reachable using request (POST, GET, …) or another type of technology such as a browser or a Swagger [19].
- Also, the DS components have been deployed in the same Virtual Machine where the MinIO data lake is located.
- Each part of the cluster has been deployed using Open-Source technology (Open-Source technologies).
- The solution developed in the project using Kubernetes has been updated to the official IDSA repository because of it was not available previously.
- The components with older versions in the TestBed have been updated to the last versions repairing bugs and connection problems. Among the new functionalities, the connectors have been updated to use the latest release developed by Fraunhofer (V8.0.2); Figure 11 shows the architecture of the latest version of the connector.

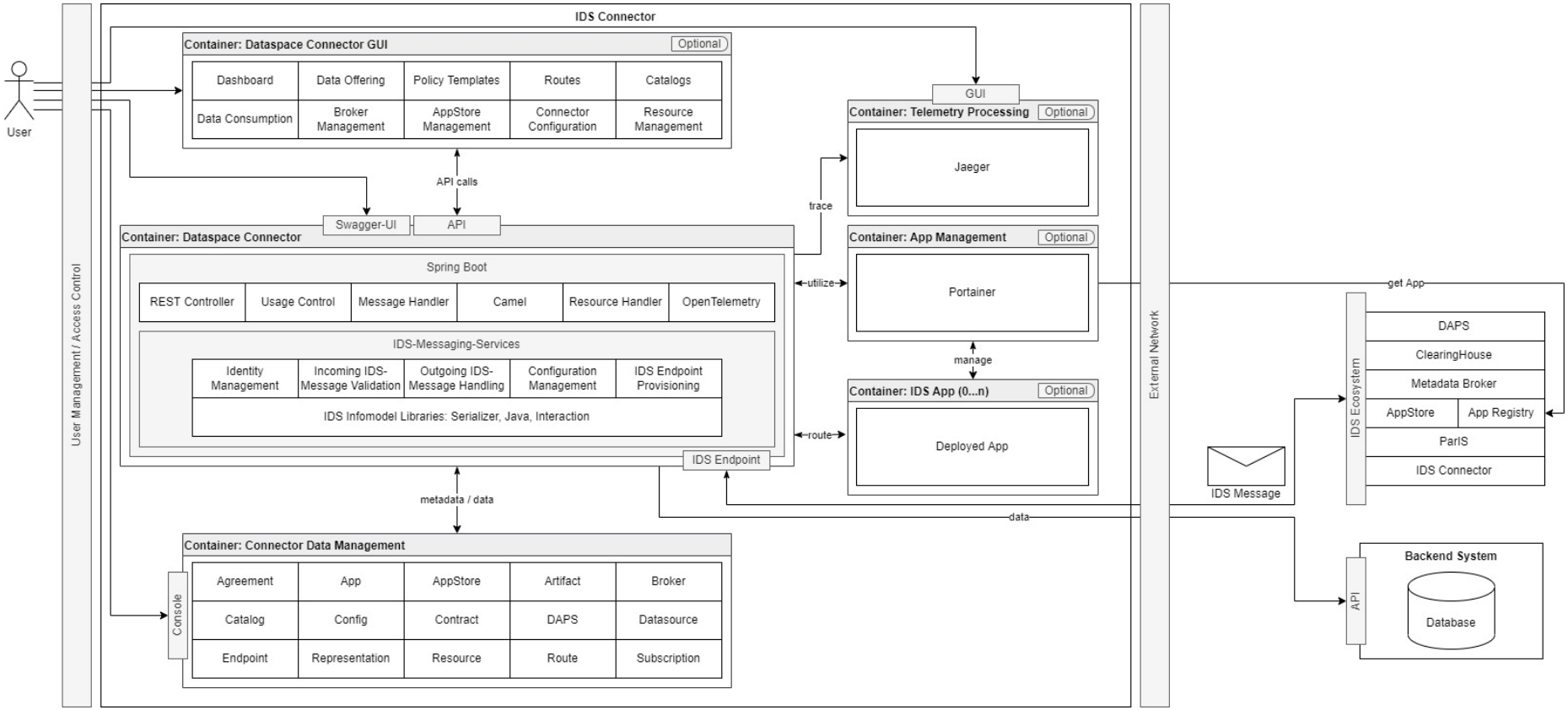

Figure 11 Data Space connector architecture. Source "https://github.com/International-Data-Spaces-Association/DataspaceConnector/blob/main/docs/assets/images/container-overview.jpg"

- The connections between the rest of the pilots and the data lake have been completed. Thus, two companies' pilots can exchange data and datasets or upload/download the dataset from MinIO data lake. Figure 12 represents this ecosystem where the core components of the DS is in the centre and the MinIO data lake is on the left side, and finally, the pilot companies are on the right side.

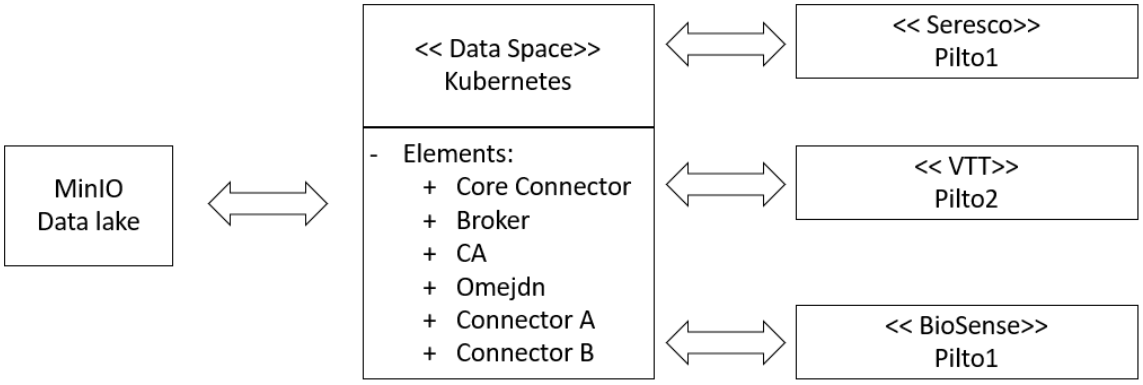

Figure 12 Data Space connected with the pilots and data lake

- The architecture shown in the Figure 13 has been designed to communicate the AI architecture with the DS. This architecture is characterised for using a REST-API based on Flask technology to create a logic communication between the AI subsystem and the rest of the elements in the platform. In addition, the REST-API will have access to a database with user credentials to ensure the security of the system. This database will be based on a structured engine, in particular on Redis [20].

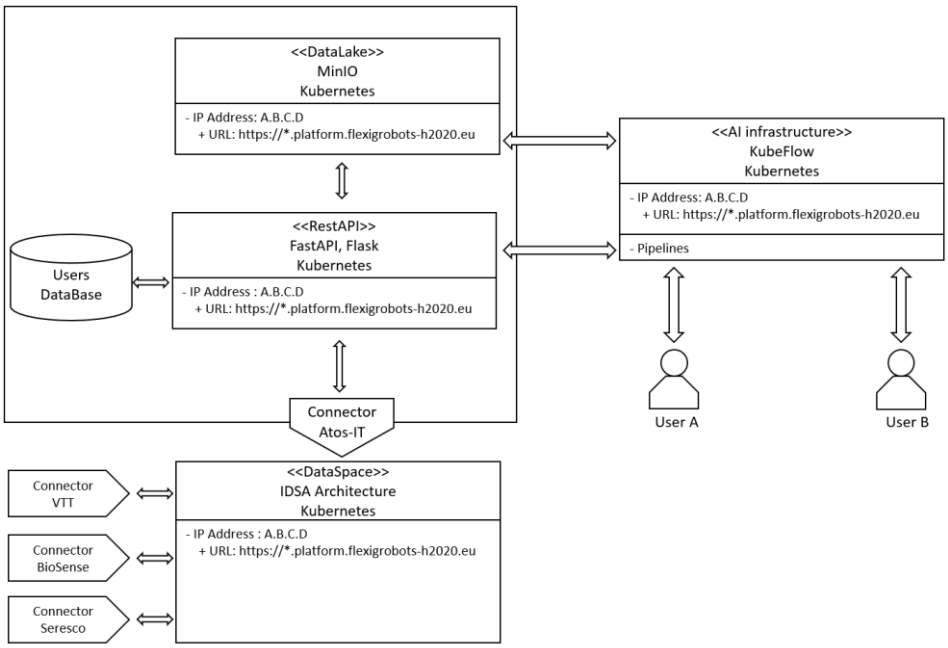

Figure 13 Communication architecture for the integration of the Data Space and the AI subsystem

3.2 Requirements

3.2.1 Technical requirements

To launch successfully an IDSA's DS is needed a set of HW and SW requirements. In this project it has been used a Virtual Machine with the next characteristics:

- Intel Xeon Processor (CascadeLake) 6 CPUs, 64bit quad-core.
- 512 GB storage.
- 12 GB RAM.
- Ubuntu 20.04.

Regarding Redis [20] database, the minimum requirements according to the official web page are:

- Redis servers need 3 or 4 vCPUs each (>2.6 Ghz) and 8GB of RAM, configured as a Redis Cluster (with high availability).
- High-bandwidth network interface card (1Gbps recommended).
- 10GB disk (to store the operating system, logs, etc.).

Regarding Flask [21] the dependencies and requirements specified in the official web page are:

- WSGI is a standard Python interface to connect severs and applications.
- Flask uses Jinja [22], this template language is used to render application in web page.
- Itsdangerous [23] is a tool to sign the data of a secure way. This tool is used for Flask to protect their sessions.

3.2.2 Functional requirements

The main aim is to exchange data between the consortium companies; in order to achieve this, firstly, the connectors have been deployed in the companies' infrastructure (cloud machines, servers, etc.). After that, the connectors must reach the DS deployed in the cloud; for this reason, it is essential to correctly configure the certificates and use the correct version of the connector (V.8.0.2).

The DS's Omejdn and the Meta Data Broker have to be exposed to the cloud, the Omejdn is the system that forms the dynamic attribute provisioning system (DAPS) intended to assure certain attributes to connectors. Therefore, third parties need not rely on the latter as long as they trust the assertions of the DAPS. And the Metadata Broker, which is a registry of self-description documents of the IDS connector. this is achieved using a Reverse proxy, in the case of this project and with the open-source mentality it has been deployed Traefik [24]. This Reverse proxy includes a dashboard that monitors the network state of the DS.

Finally, the access to the DS has been encrypted using Let's Encrypt, which is based on the use of Transport Layer Security (TLS). TLS is a protocol that, as stated in [25], "*provides security and encrypted communication between computer networks. It is the standard in multiple fields, such as HTTPS web pages, SMTP systems, or VoIP messaging; being the most used security protocol for the public*". The communication with the AI subsystem through the DS will be carried out with an API and a database. The API will use Flask, there are two main reasons to use this technology. The first one, Flask is an open-source technology and the requirements SW and HW are lower than other kind of APIs. And the second one, Flask is more flexible and customizable than other API-tools. The main aim is to link this API with three components, the AI platform explained in the previous section, with the DS and with a user database.

The database will be developed using a Redis [20] Database. The main reason to choose this database is for its in-memory store. Redis persists data to disk to ensure the durability of the data. Redis can restore from the disk whether there is a corruption or crash problem. Apart from this kind of persistence, other types of persistence methods can be adjusted for the application.

In order to integrate every component in a same technology, the API-REST and the User database will be integrated in the cluster where the DS is deployed.

3.3 Data models

FlexiGroBots project uses the information model of IDSA described in [26], this document explains in more detail the format and the sort of message. This format message is used for connectors to exchange data, Figure 14 is shown the structure of this type of message.

```
curl --location --request POST 'https://connectora.platform.flexigrobots-
h2020.eu/api/artifacts' \
--header 'Authorization: Basic XXXXX' \
--header 'Content-Type: application/json' \
--data-raw '{
    "title": "Example artifact with weather data",
    "description": "This is an for the D3.2",
    "value": "Insert data",
    "automatedDownload": true
}'
```

Figure 14 IDSA data model

To send data through this connector is used the value field in the artifact message. Hence, it has been used as Base64 binary-to-text encoding. This codification allows two things; the first one homogenizes the different data in the same language; for example, if an image is going to be sent, with Base64 encoding, the obtained data model is the same that in list values. And second, the Base64 encoding results provide an easy way to send information through API. The encoded data is attached to a POST request and sends the message through the

connector's API-REST. This allows to keep aspects described in D3.1 [14], such as supporting the description, publication and identification of data products and reusable data processing software (both referred to by the IDS-RAM as "Digital Resources" or simply "Resources"). Despite being executed in another type of architecture, such as Kubernetes, the method of sending and registering connectors is the same as when using Docker compose technology.

Figure 15 shows a real value Base64 encoding. This encoding allows passing the information bits to plain text, and the message used is a document quickly sent through a request.

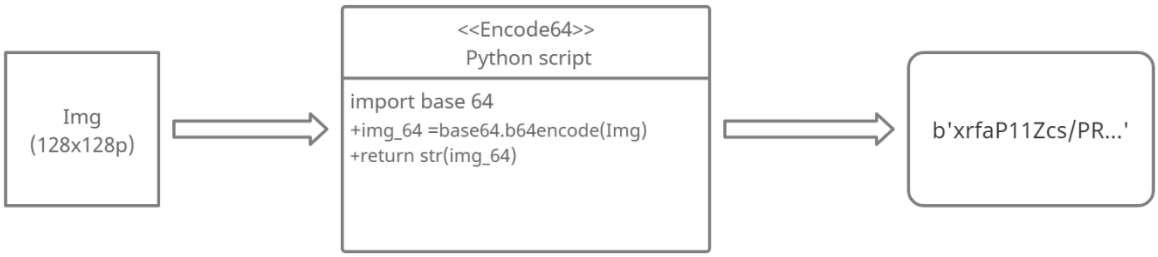

Figure 15 Example of Base64 encoding

3.4 Application Programming Interfaces (APIs)

The API connector is the same as the one stated in D3.1; the differences regarding the previous deliverable are two: the first, the connector has been updated at the last version (v8.0.2), which allows fixing bugs and connectivity problems. The second one, connectors have been exposed to the cloud; this modification allows accessing the connector using an externa URL. Figure 16 shows the connector's new version and the URL to access the API. It is worth noting that the API specifications are the same as stated in the IDSA connector description [27].

Figure 16: Dataspace Connector version 8.0.2

3.5 Graphical User Interfaces (GUIs)

Not relevant for common data services.

3.6 Installation

The steps to install the DS have been modified due to the migration from Docker to Kubernetes.

This section clarifies the order to install each DS component for the FlexiGroBots project.

1. Install Kubernetes version (1.29) or higher.
2. Install Kubectl [28].
3. Create a Kubernetes cluster to work on or join to the one assigned to the project.
4. Create a namespace to deploy the Data Space in a clean environment.
5. Launch the corresponding manifests inside the created namespace and assign the URLs to the IngressRoute that corresponds to the public IP of the machine on which the cluster is running.
6. Test the request list attached in the FlexiGroBots repository [29] and make sure the connectors are registered in the DS.

In order to install the API-REST and the Redis database a new namespace will be created in the K8S cluster. The steps to deploy them will be the same as those used for the other components: using the “apply” command in Kubectl to deploy the manifests in the cluster.

Finally, it is recommendable to connect an external connector, for this purpose it is important to have the DS’s certificates (in this project the certificates are generated for IDSA but in the future Atos will develop the CA). With these certificates it is possible to test the connector properly.

3.7 Prototype availability within FlexiGroBots

The prototype is formed by a set the systems interconnected that all of them form a DS in a real production environment:

- Firstly, the system is deployed with Kubernetes technology. Thus, the DS is running in a real production environment.
- The Kubernetes cluster has been deployed in the cloud, in a Rancher cluster. This means that any customer with a connector could register and interact with the DS.
- Each of the pilots have deployed a connector including certificates, allowing to connect their systems to the DS.
- Two Python libraries have been developed to work with the IDS connectors in an easier way and to connect this connector with the MinIO data lake.
- Currently, development tasks for the integration of the AI subsystem with the DS are being carried out. Also, these new components will be deployed in the same Kubernetes cluster where the DS is deployed on.

- Finally, the last step will be to integrate the Mission Control Centre's Data Space – which has been used for testing purposes– in the Data Space of the platform and use the DS as a link among all the platform's components.

3.8 Release planning

According to the release planning described in D3.1, User Stories ADS_US_10, ADS_US_05, ADS_US_06, ADS_US_07, ADS_US_11, ADS_US_04 have been completed. ADS_US_08, ADS_US_09, ADS_US_12, ADS_US_13, ADS_US_14, ADS_US_15 are currently in development.

User story ID	User story	Priority	Story points
ADS_US_12	Implementation of connector for geospatial services	High	2
ADS_US_13	Integration and communication of the different pilots using DS	High	3
ADS_US_14	Final and tested prototypes of the communication of the different pilots via the DS	High	13
ADS_US_08	Development of DAPS component	Medium	8
ADS_US_09	Development of CA component	Low	13
ADS_US_10	Integrate the MCC in the DS	High	8
ADS_US_11	Deploy the module to communicate the DS with the AI platform	High	13
ADS_US_15	Keep IDS-components updated with the latest releases	Low	3

Table 3 User stories for the ADS

4 Geospatial enablers and services

As described in the previous version of this deliverable (D3.1), "*the aim of the FlexiGroBots geospatial enablers and services task is to provide a set of general-purpose services and features that facilitate the access, visualization, and processing of geospatial datasets necessary for performing the daily activities on the farm*".

To that end, the main priority of this task is to provide key geospatial components planned for deployment within FlexiGroBots agricultural data space, being at its core "the provision of a data cube for facilitating for the FlexiGroBots pilots the management and access to the Copernicus satellite Sentinel 2 imagery and the drones' orthoimagery (and derived products)".

This updated version of the report describes the new implemented functionalities supporting the indexing, access and visualization of the datasets produced by the drones using the Open Data Cube (ODC).

4.1 Implemented functionalities

The provision and deployment of the data cube service –by means of the open-source implementation Open Data Cube (ODC)– has been the priority for the first release of the FlexiGroBots platform as it is the base for developing other (geospatial and/or AI-based) services for some of the pilots. This was already addressed in deliverable D3.1, and therefore, here it is being reported the progress and new functionalities implemented so far.

The setup and deployment comprehended on the one hand the **tools provided by ODC** (that is, a) Command Line Tools; b) Open Data Cube Explorer; c) Web User Interface (UI); d) Jupyter Notebooks Hub; and e) Open Geospatial Consortium (OGC) Web Services), and on the other hand, the (indexing of the) **Sentinel 2 data** necessary for the pilots' areas.

In this new version of the deliverable, it is described the indexing and offering of the pilot UAV (Unmanned Aerial Vehicle) data via the ODC. To that end, a **Python script** (i.e., stac-generator.py) has been implemented **for automatically generating the STAC (Spatio-Temporal Assets Catalogue) metadata** necessary for indexing and making aware the ODC of the new UAV datasets as they are available (in the case of the Sentinel 2 datasets provided by Amazon Web Services (AWS), the STAC metadata is already there to be indexed by the ODC).

4.2 Requirements

4.2.1 Technical requirements

The deployment and installation of the Python script ("stac-generator.py")[7] is done in the same VM where the ODC is running, therefore no additional resources are needed (please, refer to D3.1 for the technical requirements of the ODC). The script, however, has the following libraries dependencies that must be installed in the system:

- gdal-bin
- python-numpy
- boto3
- shapely
- rasterio
- rio-stac
- pystac[validation]

In addition, the script must have access to the location where the UAV data is stored (e.g., in the case of the Spanish pilot, the UAV imagery is stored in a network shared folder mounted in the VM (Virtual Machine) under "/data".).

The script is, in principle, able to generate STAC metadata for any geospatial dataset supported by GDAL[8] library, although for the case of FlexiGroBots, it is focused on raster ortho-imagery provided in GeoTiff[9] format.

4.2.2 Functional requirements

In order to generate the STAC metadata and index the UAV's imagery (as well as its derived outputs), they must be organized in a common folder structure and file naming convention (see Figure 17), so the Python script can find and handle them.

[7] available in https://github.com/FlexiGroBots-H2020/Geospatial-Enablers-Spanish-Pilot/blob/master/datacube/stac-generator.py
[8] https://gdal.org/
[9] https://en.wikipedia.org/wiki/GeoTIFF

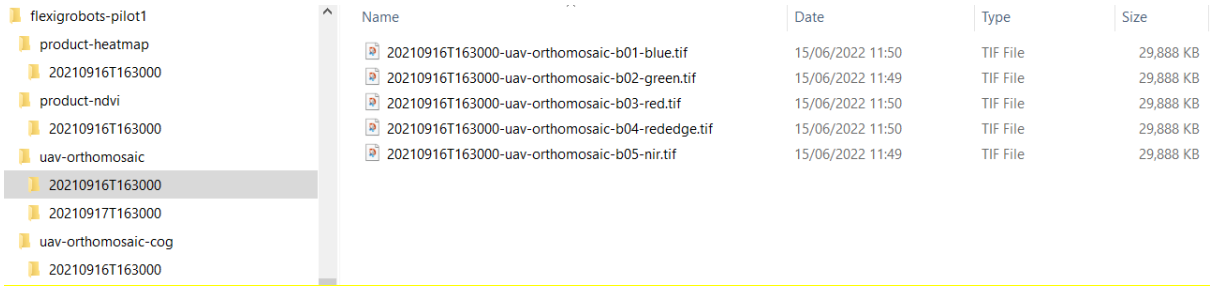

Figure 17 Folder structure and file names of the different products for various UAV flights in the Spanish pilot area

With that, the script can generate the different JSON files generated according to the STAC standard specification. That is:

- A **Catalog** that provides links to Items or to other Catalogs.
- A **Collection** (==product) that shares most fields with a Catalog, but has a few additional fields like license, extent, providers, keywords, and summaries.
- An **Item** that represents a single spatiotemporal asset, or a unit of data and metadata that contains information about the Earth captured at a certain space and time.
- An **Asset** that represents the specific metadata of a file (e.g., 20210917T163000-uav-orthomosaic-b01-blue.tif) (note: an Item can be composed of multiple assets).

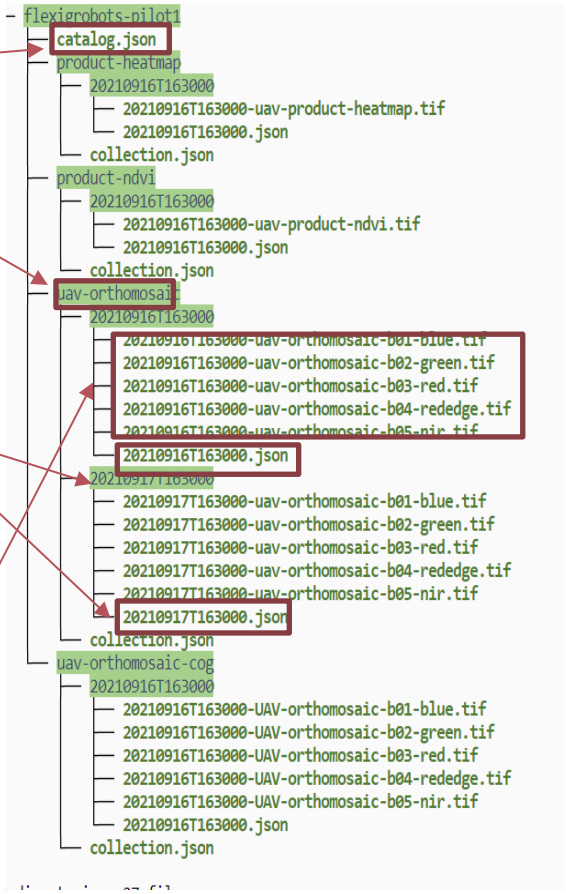

Figure 18 shows the relation of the python "stac-generator.py" script within the context of the ODC components, and how it supports the "odc-indexer" container to index the UAV's EO datasets by preparing their associated STAC metadata.

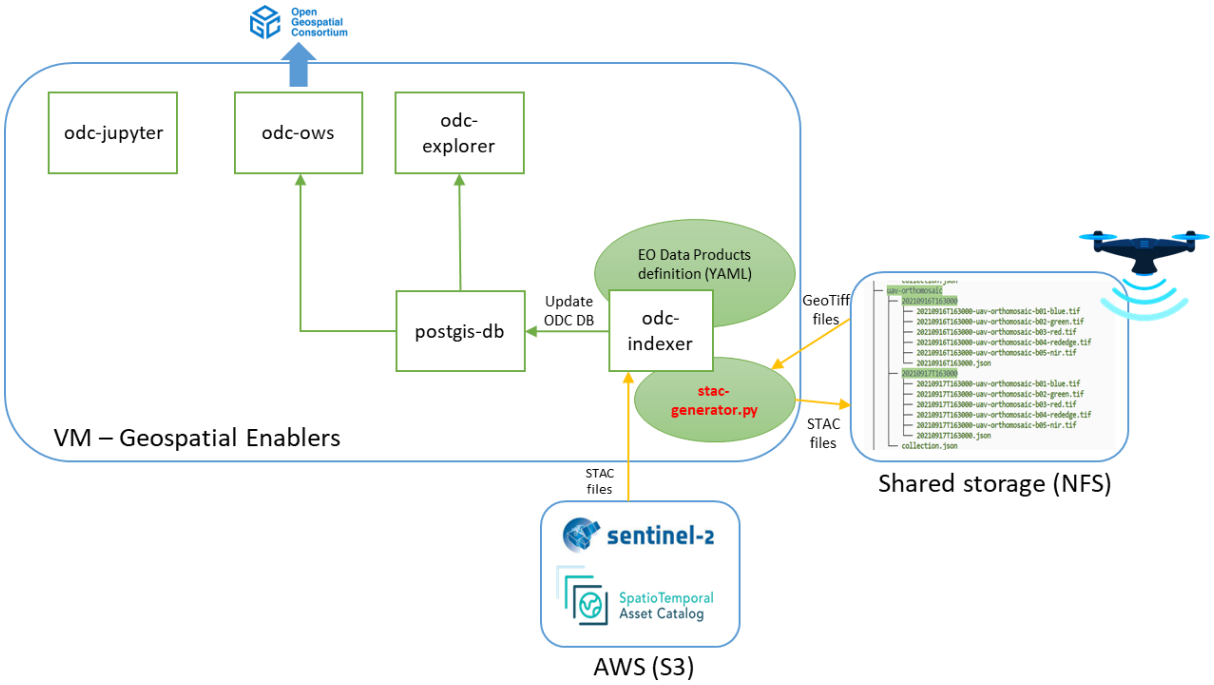

Figure 18: Relation of "stac-generator" Python script with the ODC components

4.3 Data models

The generation and indexing of the UAV datasets comprise:

- The UAV imagery and derived products, provided in GeoTIFF format
- The STAC JSON files (see previous section), generated out of the UAV imagery
- The UAV products definition, which is a YAML file containing the definition of each of the UAV outputs and derived products (one YAML file per product). This file is then used for registering the products in the ODC and linking it with the STAC JSON files.

The following are examples of the STAC JSON files produced by the Python script:

catalog.json


```
{
 "type": "Catalog",
 "id": "flexigrobots-pilot1-root-catalog",
 "stac_version": "1.0.0",
 "description": "FlexiGroBots STAC Catalog for the Spanish Pilot (Pilot #1)",
 "links": [
  {
   "rel": "root",
   "href": "http://flexi-datacube.westeurope.cloudapp.azure.com/nginx/stac/catalog.json",
   "type": "application/json",
   "title": "FlexiGroBots STAC Catalog for the Spanish Pilot (Pilot #1)"
  },
  {
```

```
    "rel": "child",
    "href": "http://flexi-datacube.westeurope.cloudapp.azure.com/nginx/stac/uav_orthomosaic/collection.json",
    "type": "application/json",
    "title": "Orthomosaics collected from regular UAV flights"
   },
   {
    "rel": "child",
    "href": "http://flexi-datacube.westeurope.cloudapp.azure.com/nginx/stac/uav_orthomosaic_cog/collection.json",
    "type": "application/json",
    "title": "Orthomosaics collected from regular UAV flights (COG version)"
   },
   {
    "rel": "child",
    "href": "http://flexi-datacube.westeurope.cloudapp.azure.com/nginx/stac/uav_ndvi/collection.json",
    "type": "application/json",
    "title": "NDVI product derived from UAV collected orthoimages"
   },
   {
    "rel": "child",
    "href": "http://flexi-datacube.westeurope.cloudapp.azure.com/nginx/stac/uav_heatmap/collection.json",
    "type": "application/json",
    "title": "Heatmap product derived from UAV collected orthoimages"
   },
   {
    "rel": "self",
    "href": "http://flexi-datacube.westeurope.cloudapp.azure.com/nginx/stac/catalog.json",
    "type": "application/json"
   }
  ],
  "stac_extensions": [],
  "properties": {},
  "title": "FlexiGroBots STAC Catalog for the Spanish Pilot (Pilot #1)"
}
```


collection.json (uav_orthomosaic)


```
{
  "type": "Collection",
  "id": "uav_orthomosaic",
  "stac_version": "1.0.0",
  "description": "Orthomosaics collected from regular UAV flights",
  "links": [
   {
    "rel": "root",
    "href": "http://flexi-datacube.westeurope.cloudapp.azure.com/nginx/stac/catalog.json",
    "type": "application/json",
    "title": "FlexiGroBots STAC Catalog for the Spanish Pilot (Pilot #1)"
```

```
  },
  {
    "rel": "item",
    "href": "http://flexi-datacube.westeurope.cloudapp.azure.com/nginx/stac/uav_orthomosaic/20210916T163000/20210916T163000.json",
    "type": "application/json"
  },
  {
    "rel": "item",
    "href": "http://flexi-datacube.westeurope.cloudapp.azure.com/nginx/stac/uav_orthomosaic/20220714T130000/20220714T130000.json",
    "type": "application/json"
  },
  {
    "rel": "item",
    "href": "http://flexi-datacube.westeurope.cloudapp.azure.com/nginx/stac/uav_orthomosaic/20220804T142500/20220804T142500.json",
    "type": "application/json"
  },
  {
    "rel": "item",
    "href": "http://flexi-datacube.westeurope.cloudapp.azure.com/nginx/stac/uav_orthomosaic/20220824T123000/20220824T123000.json",
    "type": "application/json"
  },
  {
    "rel": "item",
    "href": "http://flexi-datacube.westeurope.cloudapp.azure.com/nginx/stac/uav_orthomosaic/20220908T120000/20220908T120000.json",
    "type": "application/json"
  },
  {
    "rel": "parent",
    "href": "http://flexi-datacube.westeurope.cloudapp.azure.com/nginx/stac/catalog.json",
    "type": "application/json",
    "title": "FlexiGroBots STAC Catalog for the Spanish Pilot (Pilot #1)"
  },
  {
    "rel": "self",
    "href": "http://flexi-datacube.westeurope.cloudapp.azure.com/nginx/stac/uav_orthomosaic/collection.json",
    "type": "application/json"
  }
 ],
 "stac_extensions": [],
 "properties": {},
 "title": "Orthomosaics collected from regular UAV flights",
 "extent": {
```

```
  "spatial": {
   "bbox": [
    [
     -8.795820728356388,
     41.954311015763786,
     -8.791513991747848,
     41.958047921588744
    ]
   ]
  },
  "temporal": {
   "interval": [
    [
     "2021-09-16T16:30:00Z",
     "2022-09-08T12:00:00Z"
    ]
   ]
  }
 },
 "license": "private"
}
```


20210916T163000.json (item.json file)


```
{
 "type": "Feature",
 "stac_version": "1.0.0",
 "id": "20210916T163000",
 "properties": {
  "platform": "UAV",
  "gsd": 30,
  "proj:epsg": 25829,
  "proj:bbox": [
   516939.79761309625,
   4644723.650525363,
   517068.21438969235,
   4644873.961964584
  ],
  "proj:geometry": {
   "type": "Polygon",
   "coordinates": [
    [
     [
      516940.15538014175,
      4644723.650525363
     ],
     [
```

```
        517068.21438969235,
        4644723.957101174
       ],
       [
        517067.853918083,
        4644873.961964584
       ],
       [
        516939.79761309625,
        4644873.655387208
       ],
       [
        516940.15538014175,
        4644723.650525363
       ]
      ]
     ]
    },
    "datetime": "2021-09-16T16:30:00Z"
  },
  "geometry": {
    "type": "Polygon",
    "coordinates": [
     [
      [
       -8.795598481055004,
       41.954311015763786
      ],
      [
       -8.794053309636507,
       41.954311015763786
      ],
      [
       -8.794053309636507,
       41.95566206941497
      ],
      [
       -8.795598481055004,
       41.95566206941497
      ],
      [
       -8.795598481055004,
       41.954311015763786
      ]
     ]
    ]
  },
```

```
"links": [
 {
  "rel": "root",
  "href": "http://flexi-datacube.westeurope.cloudapp.azure.com/nginx/stac/catalog.json",
  "type": "application/json",
  "title": "FlexiGroBots STAC Catalog for the Spanish Pilot (Pilot #1)"
 },
 {
  "rel": "collection",
  "href": "http://flexi-datacube.westeurope.cloudapp.azure.com/nginx/stac/uav_orthomosaic/collection.json",
  "type": "application/json",
  "title": "Orthomosaics collected from regular UAV flights"
 },
 {
  "rel": "parent",
  "href": "http://flexi-datacube.westeurope.cloudapp.azure.com/nginx/stac/uav_orthomosaic/collection.json",
  "type": "application/json",
  "title": "Orthomosaics collected from regular UAV flights"
 },
 {
  "rel": "self",
  "href": "http://flexi-
datacube.westeurope.cloudapp.azure.com/nginx/stac/uav_orthomosaic/20210916T163000/20210916T163000.json",
  "type": "application/json"
 }
],
"assets": {
 "B01": {
  "href": "/data/flexigrobots-pilot1/uav_orthomosaic/20210916T163000/20210916T163000-uav-orthomosaic-b01-
blue.tif",
  "type": "image/tiff; application=geotiff",
  "file:size": 30605028,
  "eo:bands": [
   {
    "name": "B01",
    "common_name": "blue",
    "description": "Blue: 475 - 20 nm",
    "center_wavelength": 0.475,
    "full_width_half_max": 0.02
   }
  ],
  "proj:epsg": 25829,
  "proj:geometry": {
   "type": "Polygon",
   "coordinates": [
    [
     [
```

```
          516940.1546510186,
          4644723.956236722
        ],
        [
          517067.85465277196,
          4644723.956236722
        ],
        [
          517067.85465277196,
          4644873.656238778
        ],
        [
          516940.1546510186,
          4644873.656238778
        ],
        [
          516940.1546510186,
          4644723.956236722
        ]
      ]
    ]
  },
  "proj:bbox": [
    516940.1546510186,
    4644723.956236722,
    517067.85465277196,
    4644873.656238778
  ],
  "proj:shape": [
    2994,
    2554
  ],
  "proj:transform": [
    0.050000000686501574,
    0.0,
    516940.1546510186,
    0.0,
    -0.050000000686501574,
    4644873.656238778,
    0.0,
    0.0,
    1.0
  ],
  "raster:bands": [
    {
      "data_type": "float32",
      "scale": 1.0,
```

```
      "offset": 0.0,
      "sampling": "area",
      "statistics": {
       "mean": 7902.81689453125,
       "minimum": 0.0,
       "maximum": 65535.0,
       "stddev": 8515.0390625,
       "valid_percent": 0.00011173483981693364
      },
      "histogram": {
       "count": 11,
       "min": 0.0,
       "max": 65535.0,
       "buckets": [
        393547,
        284446,
        133561,
        55360,
        18288,
        6103,
        2099,
        865,
        382,
        325
       ]
      }
     }
    ],
    "roles": [
     "data"
    ]
   },
```


…. Edited for space saving in the document …


```
,
   "B05": {
    "href": "/data/flexigrobots-pilot1/uav_orthomosaic/20210916T163000/20210916T163000-uav-orthomosaic-b05-
nir.tif",
    "type": "image/tiff; application=geotiff",
    "file:size": 30605028,
    "eo:bands": [
     {
      "name": "B05",
      "common_name": "nir",
      "description": "Near-IR1: 840 - 40 nm",
      "center_wavelength": 0.84,
      "full_width_half_max": 0.04
```

```
    }
  ],
  "proj:epsg": 25829,
  "proj:geometry": {
    "type": "Polygon",
    "coordinates": [
      [
        [
          516940.1546510186,
          4644723.956236722
        ],
        [
          517067.85465277196,
          4644723.956236722
        ],
        [
          517067.85465277196,
          4644873.656238778
        ],
        [
          516940.1546510186,
          4644873.656238778
        ],
        [
          516940.1546510186,
          4644723.956236722
        ]
      ]
    ]
  },
  "proj:bbox": [
    516940.1546510186,
    4644723.956236722,
    517067.85465277196,
    4644873.656238778
  ],
  "proj:shape": [
    2994,
    2554
  ],
  "proj:transform": [
    0.050000000686501574,
    0.0,
    516940.1546510186,
    0.0,
    -0.050000000686501574,
    4644873.656238778,
```

```
      0.0,
      0.0,
      1.0
    ],
    "raster:bands": [
     {
      "data_type": "float32",
      "scale": 1.0,
      "offset": 0.0,
      "sampling": "area",
      "statistics": {
       "mean": 8316.51953125,
       "minimum": 0.0,
       "maximum": 65535.0,
       "stddev": 9511.4453125,
       "valid_percent": 0.00011173483981693364
      },
      "histogram": {
       "count": 11,
       "min": 0.0,
       "max": 65535.0,
       "buckets": [
        424589,
        228014,
        130094,
        66882,
        26337,
        11054,
        4804,
        2068,
        790,
        344
       ]
      }
     }
    ],
    "roles": [
     "data"
    ]
   }
  },
  "bbox": [
   -8.795598481055004,
   41.954311015763786,
   -8.794053309636507,
   41.95566206941497
  ],
```

```
 "stac_extensions": [
  "https://stac-extensions.github.io/eo/v1.0.0/schema.json",
  "https://stac-extensions.github.io/projection/v1.0.0/schema.json"
 ],
 "collection": "uav_orthomosaic"
}
```

And the YAML file with the UAV ortho-mosaic definition:
flexigrobots_pilot1_uav_orthomosaic.odc-product.yaml

```
---
name: uav_orthomosaic
description: Orthomosaics collected from regular UAV flights in the Spanish pilot geographic area
metadata_type: eo3

metadata:
 product:
  name: uav_orthomosaic

measurements:
 - name: "B01"
  aliases: [band_01, b01, blue]
  units: "1"
  dtype: float32
  nodata: 0

 - name: "B02"
  aliases: [band_02, b02, green]
  units: "1"
  dtype: float32
  nodata: 0

 - name: "B03"
  aliases: [band_03, b03, red]
  units: "1"
  dtype: float32
  nodata: 0

 - name: "B04"
  aliases: [band_04, b04, rededge, red-edge, red_edge]
  units: "1"
  dtype: float32
  nodata: 0

 - name: "B05"
  aliases: [band_05, b05, nir, near-infra-red, near_infra_red]
```

```
  units: "1"
  dtype: float32
  nodata: 0

- name: "B06"
  aliases: [band_06, b06, panchromatic]
  units: "1"
  dtype: float32
  nodata: 0

- name: "B07"
  aliases: [band_07, b07, thermal, lwir]
  units: "1"
  dtype: float32
  nodata: 0
```

4.4 Application Programming Interfaces (APIs)

The OGC APIs described in D3.1 are still relevant for the use of the Open Data Cube.

4.5 Graphical User Interfaces (GUIs)

The graphical user interfaces described in D3.1 are still relevant for the use of the Open Data Cube.

4.6 Installation

The Python script for generating the STAC metadata is available in GitHub: https://github.com/FlexiGroBots-H2020/Geospatial-Enablers-Spanish-Pilot

The complete instructions for cloning locally the GitHub project (which includes now the "stac-generator.py" script) and deploying a working ODC instance can be found in D3.1.

The following instructions focus on the requirements installation for running the script and the necessary steps for generating the STAC metadata corresponding to the UAV imagery and indexing it in the ODC.

```
1. Install the following packages with apt-get install and pip3

   $ sudo apt-get update && sudo apt-get install python3-pip python-numpy gdal-bin
libgdal-dev

   $ pip3 install boto3 shapely rasterio rio-stac pystac[validation]

2. Generate the STAC metadata
```

```
   $ cd Geospatial-Enablers-Spanish-Pilot/datacube/

   $ ./stac-generator.py

3. Index the data products in the ODC using the STAC metadata generated in the previous step and the products definitions (YAML files)
   $ docker exec -it ${CONTAINER_NAME} /bin/bash -c "fs-to-dc --stac --update-if-exists --allow-unsafe /data/flexigrobots-pilot1; exit"

4. Update the ODC Explorer so the new data is visible
   $ docker exec -it ${CONTAINER_NAME} /bin/bash -c "cubedash-gen --init --all; sleep 5; cubedash-run; sleep 5; exit"

5. Update the ODC OGC services so they can offer the new indexed data
   $ docker exec -it ${CONTAINER_NAME} /bin/bash -c "datacube-ows-update --schema --role postgres; sleep 5; datacube-ows-update --views; sleep 5; datacube-ows-update; sleep 5; exit"
```

4.7 Prototype availability within FlexiGroBots

The "stac-generation.py" script is already available and has been successfully used to index within the ODC the data produced from the various UAV flights in the Spanish pilot area.

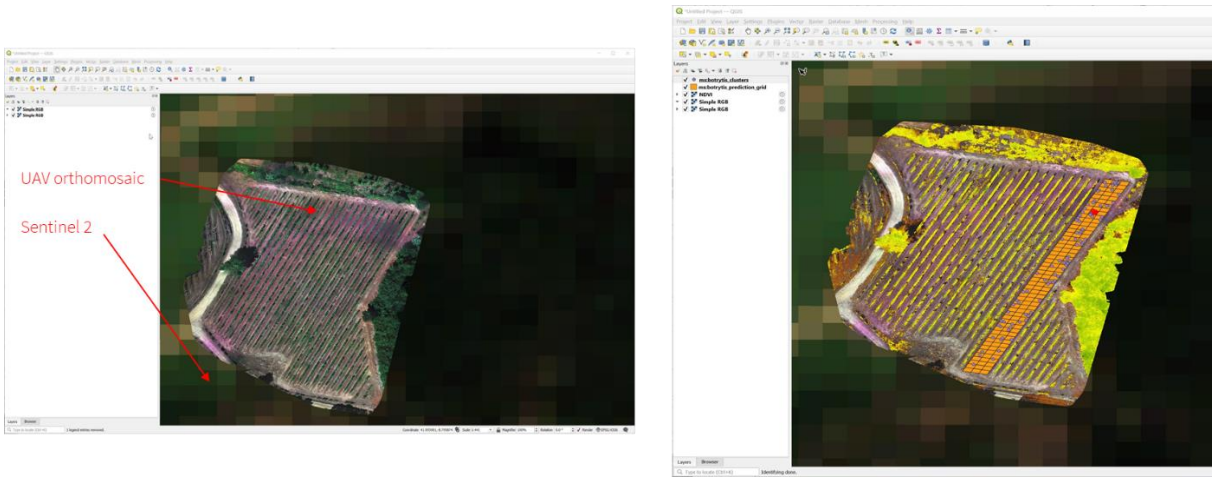

Figure 19 Visualization and integration of the different raster layers offered by ODC using the QGIS tool (images correspond to points 1 and 2 respectively)

Figure 19 shows in QGIS[10]

1) the Combined visualization of Sentinel 2 (lower resolution) and UAV ortho-mosaic (higher resolution) of the Spanish pilot area using the OGC service provided by ODC
2) the Combined visualization of a) Sentinel 2 (lower resolution), b) UAV orthomosaic (higher resolution), c) NDVI calculated from UAV orthomosaic (the calculation is done

[10] http://qgis.org/

on-the-fly by the ODC), and 4) the Botrytis vector-based datasets provided by MapServer component using OGC WFS protocol.

4.8 Release planning

According to the release planning described in D3.1, User Stories GEO_US_01, GEO_US_02, GEO_US_03, GEO_US_04, GEO_US_05 and GEO_US_06 have been completed. User Stories GEO_US_07, GEO_US_08 and GEO_US_09 are currently on hold as the pilots have not requested their implementation.

User story ID	User story	Priority	Story points
GEO_US_07	Deployment and configuration of odc-wps on top of the ODC in order to enable the possibility to directly execute specific algorithms using the EO data offered by the ODC using the OGC WPS API	Medium	6
GEO_US_08	Implement algorithms (to be discussed and prioritized with pilots, e.g., NDVI calculation) using ODC EO data and expose them via the OGC WPS API	Medium	8
GEO_US_09	Index other additional EO/raster-based datasets potentially useful/necessary for the FlexiGroBots pilots (e.g., Landsat 8 imagery, DEM, etc.)	Low	4

Table 4 User stories for the geospatial processing and services

5 Common application services

As introduced in deliverable D3.1, and as a conclusion of the requirements gathering carried out at the beginning of the project, there are several needs common among FlexiGroBots' pilots. These needs are the basis for the development of the common applications within the context of task T3.4, whose progress from the state presented in D3.1 will occupy the following pages of this document.

As a brief recap, the common applications are grouped into 3 clusters: (i) Situational Awareness, (ii) Utilities, and (iii) Generalisation. These software applications are intended to be reusable and can be easily customized to solve pilot-specific tasks. For further information on each cluster, please refer to the appropriate category below.

The implementation of these applications has included state-of-the-art architectures, datasets, repositories, and models. The combination and adaptation of all these elements, together with in-house developments and data collected in the field, represent the real added value of FlexiGroBots' T3.4 task.

5.1 Situational Awareness

Major developments have been included, improved, and updated in this group of tools since the status reported in D3.1.

It is recalled that the aim of these tools is to provide robots and their operators with as much relevant information as possible about their operational environment. The tools included into this group are: (i) SLAM, (ii) People detection, location, and tracking, (iii) People behaviour estimation, and (iv) Moving objects detection.

5.1.1 SLAM

Recapping on D3.1, the objective remains the same, to reconstruct the environment in which a robot navigates and to locate the robot in this environment, but the approach has changed. The fundamental idea is still to do it based on the visual features of the environment, but the disappointing results obtained by applying state-of-the-art algorithms [30] [31] on video captured with monocular camera have led the development team to consider alternative solutions. A sample of the results obtained, and their interpretation can be seen on Figure 20 and in the following paragraph.

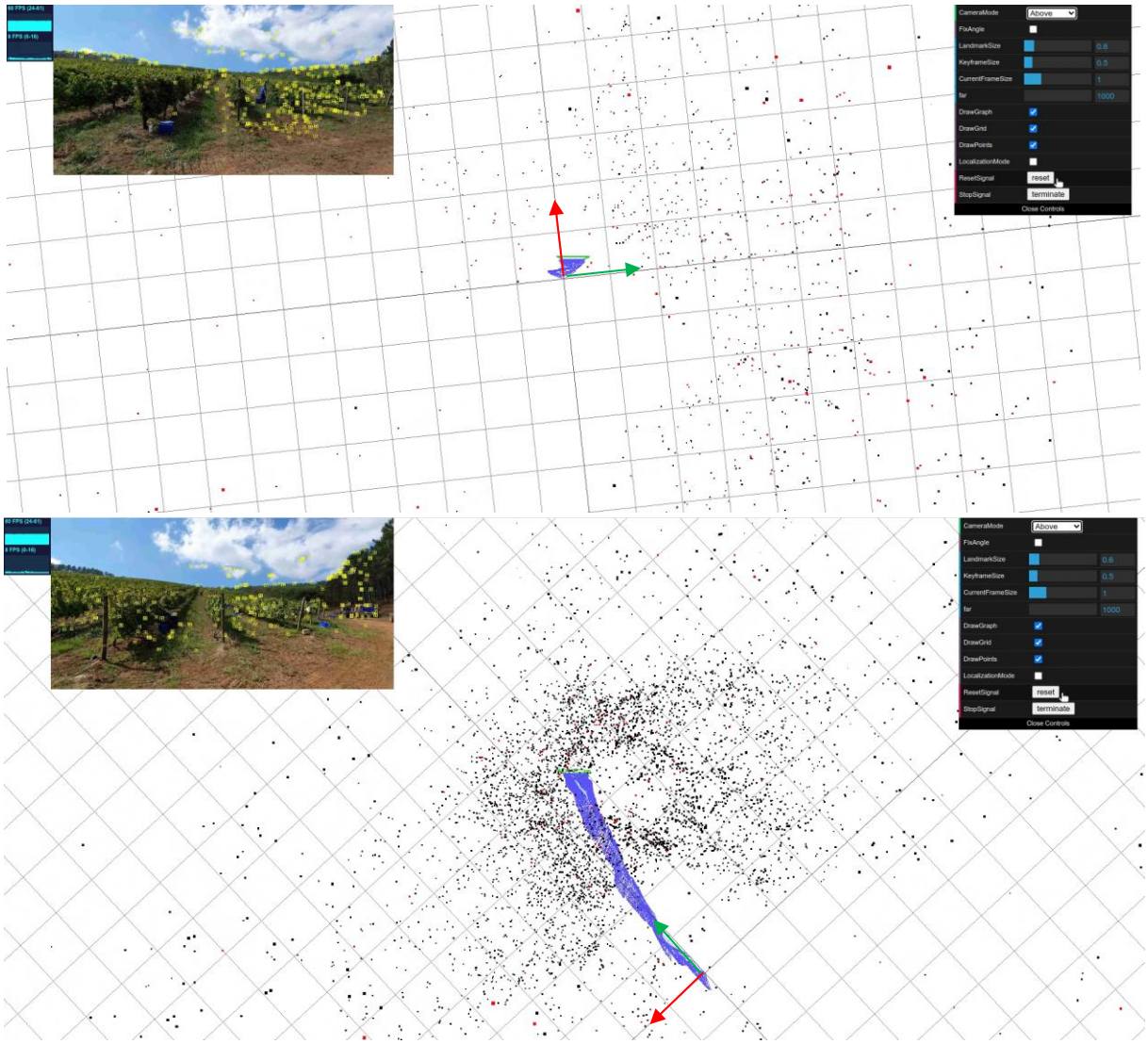

Figure 20 Frames resulting from the application of the algorithm [32] on video from Pilot 1

These two frames attached above reflect the malfunctioning of the previously, D3.1, proposed SLAM approach, using [32] as the implementation of [30]. The two frames are close in time, the real displacement of the camera is horizontal with constant orientation, but the pose estimated by the SLAM algorithm reflects a non-existent twist of about 110 degrees.

For this reason, it has been decided to introduce the use of Zed2i [33] , a stereoscopic camera with inertial info, into the project. The new approach is to adapt the commercial product's own software to the needs of FlexiGroBots' pilots, adding the necessary extra functionalities in case of detecting possible improvements.

The tests being carried out with the camera and commercial software are still at an early stage, but first results have already been obtained in tests in an indoor scenario, Figure 21, and efforts continue to make the most of it and adapt this technology for the FlexiGroBots' use case.

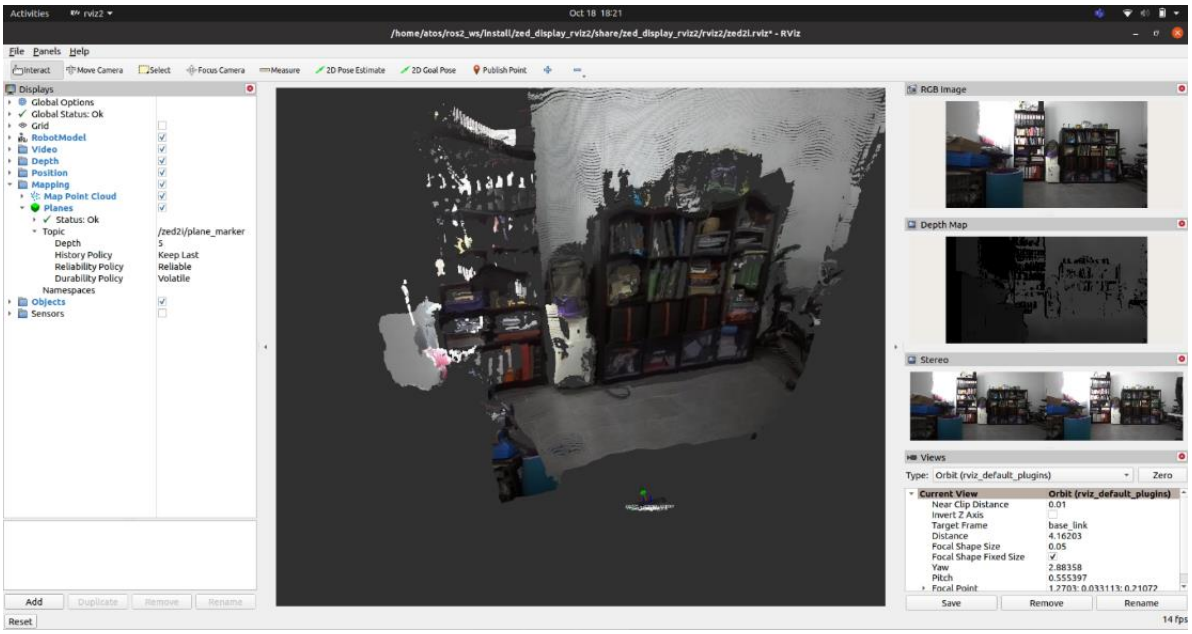

Figure 21 GUI of pose estimation and 3D reconstruction in indoor scenario with stereo camera [33]

The main added value of such an application for the project is to be able to locate the robots and estimate their trajectory even when GPS signal is lost or not available, as well as to have a 3D reconstruction of the scenarios that can be useful to inspect the field in a virtual way or to keep a historical record of the state of the crop/planting.

5.1.1.1 Implemented functionalities

After discarding the first solution implemented due to poor performance, the solution currently proposed is to use the software offered by the manufacturers of the stereo camera [33], which has the necessary functionalities already implemented, such as:

- 3D positional tracking and pose of the camera.
- Spatial mapping using visual-inertial SLAM technology.

Work now focuses on adapting these functionalities and testing their usefulness in the context of the project.

5.1.1.2 Technical requirements

The main technical requirement that has been modified since the previous release is the replacement of the monocular camera by a stereoscopic one.

All technical requirements for using this commercial camera and its SDK are defined on the vendor's website [33]. The SDK can be downloaded on Linux, Windows, and Jetson; and offers solutions for both Python and C++ languages. It is necessary to have sufficient disk space to be able to download and run the docker image, also enough ram to host models. It is highly recommended to have a Nvidia GPU to speed up the auto-labelling process of the data.

5.1.1.3 Functional requirements

Not new functional requirements included from them introduced in D3.1.

5.1.1.4 Data models

Using the stereo camera as input, this application generates an incremental 3D map, which is stored as a point cloud in a fpc.obj file or as a 3D mesh in mesh_gen.obj, and provides the output streams of the individual cameras, the composite image and the depth estimation.

5.1.1.5 Application Programming Interfaces (APIs)

The commercial product [33] provides its own python API that is being used as interface to obtain the pose and spatial information. All components listed in D3.1 continue to apply.

5.1.1.6 Graphical User Interfaces (GUIs)

Several graphical user interfaces are included in the commercial tool to facilitate the correct visualisation of the included functionalities. As an example, see Figure 21 attached above, and Figure 22 extracted from the camera documentation.

Figure 22 3D positional tracking and mapping with Zed2i stereo cam. Source: [33]

5.1.1.7 Installation

The installation steps are included as part of the material provided on the manufacturer's website [33]. If it's necessary, further information will be given in the requirements file attached with the code.

5.1.1.8 Prototype availability within FlexiGroBots

This module has not yet reached the stage of functional prototype available to all stakeholders on the AI platforms or the project repository. The change of solution has caused this tool to fall behind the estimated status.

5.1.1.9 Release planning

User story ID	User story	Priority	Story points
CAS_SLAM_US_05	Create virtual maps and 3D meshes from Pilot 1 and Pilot 3 scenes	Medium	5
CAS_SLAM_US_06	Upload the application to the AI platform and prepare deployment for required devices	Medium	3
CAS_SLAM_US_07	Adapt stereo cam 3D position and pose estimation code to project's context	High	8
CAS_SLAM_US_08	Adapt stereo cam spatial mapping code to project's context	High	8

Table 5 User stories for SLAM

5.1.2 People detection, location, and tracking

There have been no changes in the definition of the tool and its functional requirements with respect to D3.1. It is still intended to use a monocular camera mounted on the UGV to monitor the robot's environment and provide it with a degree of situational awareness of visible people and objects.

Extensive software development has been carried out for this application, with new models for detection, segmentation, tracking and depth estimation having been introduced to adopt existing major advances since the emergence of visual transformers as a disruptive model architecture for computer vision AI solutions.

Figure 23 shows the result of the application in two time instants collected in the same video of Pilot 1. On the left are the detections with their track id, together with an asterisk indicating the nearest person and a label indicating whether the person is at a safe distance or whether he/she is in a caution zone. On the right is the estimated depth of each of the scenes from an image collected with a monocular camera.

Figure 23. Output frames from PDLT app: RGB with detection info (left), monocular depth estimation (right)

The currently designed algorithm uses Detic, proposed in [34], as the detection and segmentation model. This model takes advantage of CLIP's [35] ability to relate textual and visual concepts, so that Detic can be trained for detection tasks using datasets prepared for classification tasks. This is a huge evolution, as classification datasets have a larger number of labelled concepts. In addition, the very fact of adopting CLIP allows Detic to perform zero-shot detection on unseen objects during model training.

The tracking of people and objects is carried out by combining the output of the previous detector with a particular implementation of DeepSORT [36] that uses CLIP as visual feature extractor. This implementation, adopted from [37], replaces the conventional re-ID part of DeepSORT to make it applicable to all types of objects without the need for retraining, as the convolutional model included in the standard DeepSORT is optimised for tracking people.

The estimation of the depth of objects and people is carried out by overlapping the segmentation mask obtained with Detic on the output of a visual transformer trained to estimate the depth of scenes captured with a monocular camera, DPT [38]. As shown in Figure 24, the edges of the segmentation masks are eroded to avoid picking up parts of the background, and the pixel values of that central area of the objects are averaged. The output image of the depth estimation model is a single channel image, in which each pixel represents an estimate of the absolute depth of the element to which that pixel belongs.

Figure 24. Segmentation and erode process to estimate mean object depth

In order to improve the correspondence between the depth estimates of consecutive frames, Kalman filters have been introduced to smooth the values according to the previous ones. To obtain more accurate estimation of the position of the elements in the camera's field of view, without resorting to the SLAM technology introduced in the previous section 5.1.1, it would be necessary to perform a complex calibration, not of the camera intrinsic, but of the depth estimation algorithm. For this reason, it is considered that the objective of increasing safety is solved by providing the robot with a relative knowledge of the position of certain elements and being able to establish proximity limits to avoid collisions.

The entire pipeline described above is shown in Figure 25.

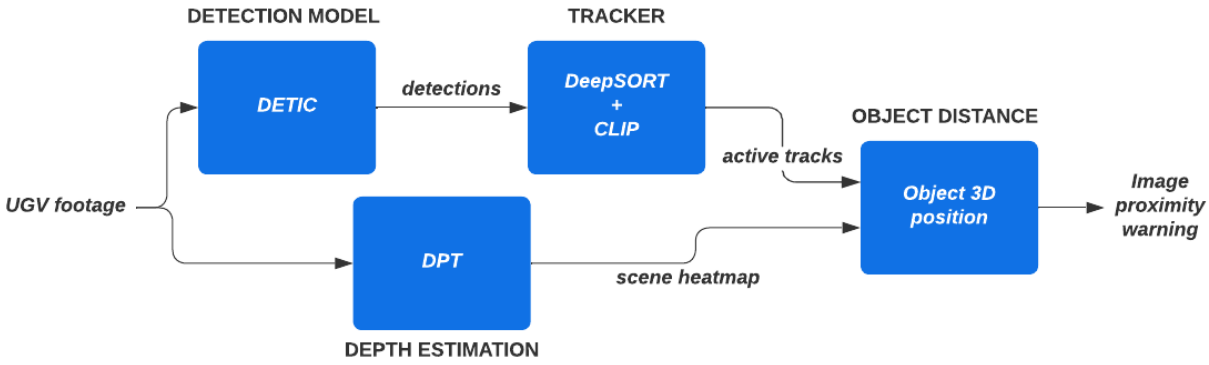

Figure 25 PDLT pipeline description

The main added value of such an application for the project is to increase awareness of the robot's environment and thus improve safety, without the need for expensive stereo cameras.

The current software not only covers the requirements described for the task but goes further and extends these functionalities not only for people, but also for other objects or vehicles that may be present in an agricultural field.

5.1.2.1 Implemented functionalities

The current implementation of the tool offers the following functionalities:

- Detection and tracking of people.
- Detection and tracking of other objects or vehicles.

- Approximate estimation of the distance of persons/objects from the monocular camera. Kalman filter included to smooth the frame-to-frame depth estimation.
- Visual warning in case of getting too close to the camera position. Settable threshold safety distance.
- Visual marking of the nearest person/object by marking *.

5.1.2.2 Technical requirements

All requirements defined in D3.1 remain valid. The implemented models are HW demanding, requiring a powerful GPU to give an adequate fps performance, between 3 and 5 depending on the number of objects tracked, in a Nvidia 3080 family GPU. Models need at least 6 GB of free GPU memory to be allocated, and docker image consume up to 15 GB of disk. This tool has been also tested in a Nvidia GeForce GTX 1070 with CUDA 11.3 and python3.8.

5.1.2.3 Functional requirements

There have been no variations with respect to what was specified in the previous document, D3.1.

5.1.2.4 Data models

This application does not store data in any format, it takes a video stream as input and provides one or more video streams as output, in mp4/avi format, or individual frames, in jpg/png. There is the possibility of defining other outputs, such as proximity alerts, but at this point of the project the requirements and formats of these messages have not been defined.

5.1.2.5 Application Programming Interfaces (APIs)

The definition of a final interface to interact with the tool is pending the implementation of the tool as an inference service in the FlexiGroBots AI platform. Currently, the software is executed as a python script inside a docker container. All components listed in D3.1 continue to apply.

5.1.2.6 Graphical User Interfaces (GUIs)

There is no GUI beyond the display of the output videos themselves.

5.1.2.7 Installation

To install this application and be able to run it, it is only necessary to have the required permissions in the repository of the project to download the app docker image, as following:

```
docker pull ghcr.io/flexigrobots-h2020/pdlt-tool
```

It is possible to check the possible input parameters of the application script through the following command:

```
docker run -it pdlt-tool --help
```

Once you have checked that the image is running properly and the possible input parameters are known, delete the previous container and run it again with the desired parameters.

5.1.2.8 Prototype availability within FlexiGroBots

The prototype of this application is available in the project's GitHub repository by downloading and executing the docker image named *pdlt-tool*.

Access to the code repository is private, only members of the repository are allowed. People linked to the project who are not members of the repository will be able to access on demand. The link to the private repository is: https://github.com/FlexiGroBots-H2020/pdlt-tool.

5.1.2.9 Release planning

User story ID	User story	Priority	Story points
CAS_PLDT_US_05	Evaluate and test in field conditions (Pilots 1 and 3)	Medium	3
CAS_PLDT_US_06	Upload the application to the AI platform and prepare deployment for required devices	Medium	3
CAS_PLDT_US_07	Refine scene depth estimation	High	2
CAS_PLDT_US_08	Define and implement possible communications with the MCC	Low	2

Table 6 User stories for people detection, location, and mapping

5.1.3 People action recognition

The objectives and task description reflected in D3.1 are maintained, but the software has been completely updated since that release. The current solution implements a spatiotemporal action detection model [39] trained on the AVA dataset [40].

AVA actions dataset has 80 labelled actions, among which several of interest for the context of the FlexiGroBots project can be found. The images included in Figure 26 show frames extracted from videos processed with this common application and they demonstrate how it is possible to detect actions such as standing, crouching, kneeling, walking, or carrying something.

Figure 26 Human action recognition application applies to raw images from Pilot 1

The added value of detecting this type of actions is to increase safety of the operators, for example, being able to detect people sitting or lying on the floor, or people walking too close to the robot if this AI module is linked with the previous tool module. There are a multitude of use cases that could use this tool to obtain valuable information from the scene.

5.1.3.1 Implemented functionalities

The current implementation of the tool offers the following functionalities:

- People detection in image.
- Human action recognition based on the spatiotemporal evolution of their body position. 80 classes of actions available in the original implementation [39].

5.1.3.2 Technical requirements

This tool has been implemented in a Nvidia GeForce GTX 1070 with CUDA 11.3 and python3.8, achieving frame rates of 5 fps and requesting 6 GB of free GPU memory to host the model. Docker image needs 12 GB of free space in memory.

5.1.3.3 Functional requirements

There have been no variations with respect to what was specified in the previous document, D3.1.

5.1.3.4 Data models

This application does not store data in any format, it takes a video stream as input and provides one or more video streams as output, in mp4/avi format, or individual frames, in jpg/png. There is the possibility of defining other outputs, such as alert to certain recognized actions, but at this point of the project the requirements and formats of these messages have not been defined.

5.1.3.5 Application Programming Interfaces (APIs)

The definition of a final interface to interact with the tool is pending the implementation of the tool as an inference service in the FlexiGroBots AI platform. Currently, the software is executed as a python script inside a docker container. All components listed in D3.1 continue to apply.

5.1.3.6 Graphical User Interfaces (GUIs)

There is no GUI beyond the display of the output videos themselves.

5.1.3.7 Installation

To install this application and be able to run it, it is only necessary to have the required permissions in the repository of the project to download the app docker image, as following:

```
docker pull ghcr.io/flexigrobots-h2020/har-tool
```

It is possible to check the possible input parameters of the application script through the following command:

```
docker run -it har-tool --help
```

Once you have checked that the image is running properly and the possible input parameters are known, delete the previous container and run it again with the desired parameters.

5.1.3.8 Prototype availability within FlexiGroBots

The prototype of this application is available in the project's GitHub repository by downloading and executing the docker image named *har-tool*.

Access to the code repository is private, only members of the repository are allowed. People linked to the project who are not members of the repository will be able to access on demand. The link to the private repository is: https://github.com/FlexiGroBots-H2020/har-tool.

5.1.3.9 Release planning

User story ID	User story	Priority	Story points
CAS_PBE_US_04	Evaluate and test in field conditions (Pilots 1 and 3)	Medium	3
CAS_PBE_US_05	Upload the application to the AI platform and prepare deployment for required devices	Medium	3
CAS_PBE_US_06	Adapt the implemented model for the most relevant classes of actions in the project scenarios	Medium	5
CAS_PBE_US_07	Define and implement possible communications with the MCC	Low	2

Table 7 User stories for people action recognition

5.1.4 Moving objects detection, location, and tracking

The description of the purpose and operating conditions of this application remain as described in document D3.1.

The proposed solution implements an aerial image object detection model, TPH-YOLOv5, trained with a dataset developed as part of the task, which includes images of tractors from UAVs in Pilot 1, images obtained with the tool introduced in 5.2.3, and images from the Visdrone dataset [41].

TPH-YOLOv5 is a variation of the YOLOv5 [42] model, widely used in image object detection applications, specifically modified to improve performance in aerial scenarios where objects have tiny sizes. This model has four Transformer Prediction Heads (TPH) that connect to the convolutional backbone of the conventional YOLO architecture, thus incorporating the attention mechanism of the Transformers to improve performance in distant images. More model details can be found in Annex A.

The tracking of the detected objects is performed by the algorithm introduced in section 5.1.2, an adaptation of DeepSort with CLIP as feature extractor [37]. For more details, it is recommended to read the contents of the above-mentioned section.

Figure 27 schematically represents the processing flow of the proposed solution.

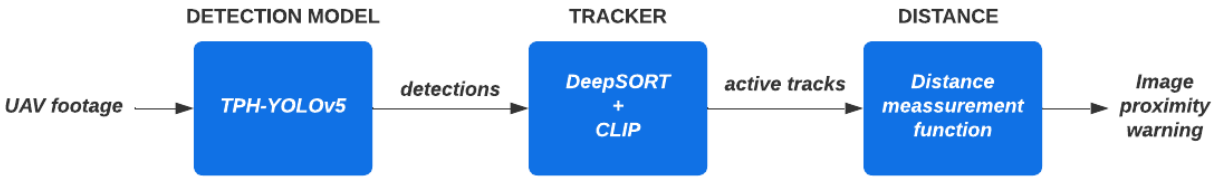

Figure 27 MODLT pipeline description

Figure 28 shows some frames that exemplify the results obtained by applying the introduced detection and tracking tool. The trained model can detect and identify different concepts, such as tractors, cars, vans, and people. More dataset details can be found in Annex A.

Figure 28 Output frames from MODLT common application pipeline

The added value of this application is the increased monitoring and safety in the agricultural field. Having the ability to monitor the position and trajectory of vehicles and people means being able to foresee possible risk situations in which the actors get too close to each other or move away from predefined working area. This tool will be very useful for the mission control centre and for adding an extra layer of safety to autonomous tractors.

5.1.4.1 Implemented functionalities

The current implementation of the tool offers the following functionalities:

- Detection of objects in aerial images taken with UAV from heights up to 100 meters. Detected classes are tractor, people, car, van, and truck.
- Tracking of detected objects.

5.1.4.2 Technical requirements

This tool has been implemented in a Nvidia GeForce GTX 1070 with CUDA 11.3 and python3.8, achieving frame rates of 5 fps and requesting 4 GB of free GPU memory to host the models. Docker images needs 15 GB of free space in memory.

5.1.4.3 Functional requirements

There have been no variations with respect to what was specified in the previous document, D3.1.

5.1.4.4 Data models

This application does not store data in any format, it takes a video stream as input and provides one or more video streams as output, in mp4/avi format, or individual frames, in jpg/png. There is the possibility of defining other outputs, such as certain events alert, but at this point of the project the requirements and formats of these messages have not been defined.

5.1.4.5 Application Programming Interfaces (APIs)

The definition of a final interface to interact with the tool is pending the implementation of the tool as an inference service in the FlexiGroBots AI platform. Currently, the software is executed as a python script inside a docker container. All components listed in D3.1 continue to apply.

5.1.4.6 Graphical User Interfaces (GUIs)

There is no GUI beyond the display of the output videos themselves. in the future, if this tool is included in the mission control centre, its GUI will be used to display the information.

5.1.4.7 Installation

To install this application and be able to run it, it is only necessary to have the required permissions in the repository of the project to download the app docker image, as following:

```
docker pull ghcr.io/flexigrobots-h2020/modlt-tool
```

It is possible to check the possible input parameters of the application script through the following command:

```
docker run -it modlt-tool --help
```

Once you have checked that the image is running properly and the possible input parameters are known, delete the previous container and run it again with the desired parameters.

5.1.4.8 Prototype availability within FlexiGroBots

The prototype of this application is available in the project's GitHub repository by downloading and executing the docker image named *modlt-tool*.

Access to the code repository is private, only members of the repository are allowed. People linked to the project who are not members of the repository will be able to access on demand. The link to the private repository is: https://github.com/FlexiGroBots-H2020/modlt-tool.

5.1.4.9 Release planning

User story ID	User story	Priority	Story points
CAS_MODLT_US_03	Evaluate and test in field conditions (Pilots 1, 2 and 3)	High	3
CAS_MODLT_US_04	Upload the application as AI platform inference service and prepare deployment for required devices	Medium	3
CAS_MODLT_US_05	Improve model performance by adding more data to the dataset and improving model quality	High	5
CAS_MODLT_US_06	Implement a method for estimating the distance between objects in the image to warn of possible risk situations	Medium	5
CAS_MODLT_US_07	Define and implement possible communications with the MCC	Low	2

Table 8 User stories for moving objects detection and tracking

5.2 Utilities

As introduced in D3.1, the applications included in this group are those that can be used as an added step in different processes, or as a basis for generating a specific solution for each pilot according to its own requirements.

The evolution of this group of applications since the previous release has been remarkable. Three new tools have been introduced and implemented since last deliverable, one of them replacing the “GIS plug-in” application included in D3.1.

5.2.1 Orthomosaic Assessment Tool

The application described in this section replaces the one that in document D3.1 occupied item 5.2.1, "GIS plug-in". The substitution of this application is motivated by the fact that the interest in the previous tool was not common, it was only required by a single pilot, and on the contrary, there was consensus with the one currently included. All the parties involved agreed with the modification and the content of the previous app was dumped into task T3.3.

The orthomosaic assessment tool is a software designed to generate an orthomosaic by stitching in the raw images as extracted from the UAV. Moreover, it generates the Digital Terrain Model (DTM) and Digital Surface Model (DSM) while processing the orthomosaic. This process is done by implementing the command line toolkit provided by OpenDroneMap [43].

After the generation of the primary products, the software continues running to generate intermediate products, such as the Canopy Height Model (CHM), the Normalized Difference Vegetation Index (NDVI), and the Leaf Area Index (LAI) based on the projected shadows [44]. These intermediate products are used along with ground truth data to train a Random Forest (RF) algorithm, which will predict the presence/absence of a disease based on the variables inputted. Finally, the risk points are used to generate a heatmap and a risk assessment report.

The application for which the orthomosaic assessment tool was designed was to assess the early presence of Botrytis cinerea in vineyards and hence, the final heatmap represents the hotspot areas in which the disease has higher probability to be developed.

5.2.1.1 Implemented functionalities

The current implementation of the tool offers the following functionalities:

- Detection of Botrytis cinerea hotspots using UAV multispectral imagery.
- The tool generates a Botrytis risk heatmap and an assessment report.

5.2.1.2 Technical requirements

The libraries necessary for the utilization of the orthomosaic assessment tool software are listed in the requirements.txt, included along with the code, some of the most noteworthy are Fiona, rasterio, osgeo, GDAL and geopandas. Furthermore, this application requires sufficient disk space to be able to deploy the docker container, up to 8 GB of free memory.

5.2.1.3 Functional requirements

The functional requirements defined for this tool to be able to provide a solution to the detected needs are that the user can enter raw data extracted from the UAV and that heatmaps can be generated without errors at least every 12 hours.

5.2.1.4 Data models

This application does not store data in any format, it intakes multispectral images as extracted from the UAV, process them generating intermediate products, which are stored, and finally generates the Botrytis risk heatmap and the PDF report.

5.2.1.5 Application Programming Interfaces (APIs)

The software can be executed in script mode or cross-platform inside a Python interpreter, such as PyCharm (JetBrains s.r.o., Prague, Czech Republic).

5.2.1.6 Graphical User Interfaces (GUIs)

There is no GUI beyond the display of the botrytis risk heatmap and the PDF report. For the visualization of the map, GIS software such as the open-source software called QGIS is recommended.

5.2.1.7 Installation

The user should have the software Docker [45] installed.

Next, the user should modify the directories in the main.py script to match the directory in which the raw images as extracted from the UAV are located. Moreover, the user should modify the director of the intermediate products and final deliverables.

5.2.1.8 Prototype availability within FlexiGroBots

The orthomosaic assessment tool will be available at the GitHub repository of FlexiGroBots along with a README.md file with the most relevant information to run the software.

Access to the code repository is private, only members of the repository are allowed. People linked to the project who are not members of the repository will be able to access on demand. The link to the private repository is: https://github.com/FlexiGroBots-H2020/orthomosaic-assessment-tool.

5.2.1.9 Release planning

User story ID	User story	Priority	Story points
CAS_OAT_US_01	Generate an orthomosaic, DTM, and DSM	High	8
CAS_OAT_US_02	Compute the CHM	Medium	3
CAS_OAT_US_03	Compute the NDVI and crop it to the extension of the CHM	Medium	3
CAS_OAT_US_04	Compute the LAI	Medium	3
CAS_OAT_US_05	Train the Random Forest model	High	8
CAS_OAT_US_06	Test the RF model	Medium	5
CAS_OAT_US_07	Generate the heatmap	High	5
CAS_OAT_US_08	Generate the risk report	Medium	3

Table 9 User stories for GIS plug-in

5.2.2 Anonymization Tool

The anonymization common application is introduced for the first time in this document, as it was not presented in the previous deliverable, D3.1.

The purpose of this software is to provide a solution for compliance with the General Data Protection Regulation (GDPR). In any situation that involves working with videos or images in which people's faces can be seen, problems arise with this directive, as their privacy is violated. Without this tool, explicit authorization is required to show or use these images, which is a complication and can be very limiting.

For this reason, a tool has been developed to anonymize the faces that appear in the images of the pilots, either by superimposing a black square on the face, or by generating a synthetic face using a Generative Adversarial Network (GAN). Two examples of the results obtained are shown in Figure 29 The black square approach is faster, but the fake face generation it is better to avoid bias if the output images will be used to be included in a dataset.

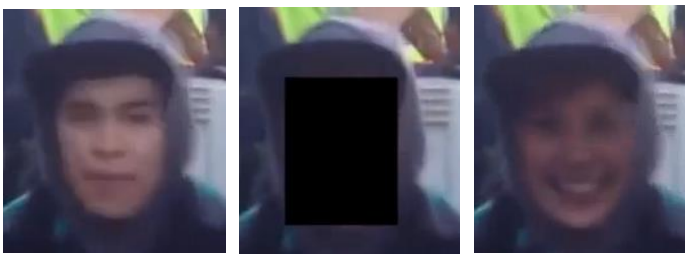

Figure 29. Anonymized faces modes in FlexiGroBots' common application

The developed tool implements the code of the DeepPrivacy repository, [46], adapting it and adding the additional functionalities needed to meet the requirements of FlexiGroBots. The solution proposed in [46] implements a two-step approach, in which first the faces and their keypoints are detected with convolutional neural networks, and then fed to GAN to generate a face in the correct position and orientation. Figure 30 shows the quality and consistency of the generated faces.

Figure 30 Fake face generation with DeepPrivacy GAN [46]

The added value of this application is not negligible, since it covers an existing need and solves possible legal issues, improving the quality of the possible datasets generated with the images collected in the pilots, avoiding a significant bias due to the presence of black or blurred squares instead of faces.

5.2.2.1 Implemented functionalities

The current implementation of the tool offers the following functionalities:

- Detect faces position in image.
- Extract face keypoints for each face in the image.
- Anonymize faces with a black box or a coherent fake face generate with a GAN.

5.2.2.2 Technical requirements

This application requires sufficient disk space to be able to deploy a docker container of the image available in the repository, 12 GB approximately, and GPU memory space to host the face detection, face key-point extraction and GAN models, up to 5 GB. It is highly recommended that the device on which the solution is deployed has an Nvidia GPU to speed

up the anonymisation process. This tool has been implemented in a Nvidia GeForce GTX 1070 with CUDA 11.3 and python3.8.

5.2.2.3 Functional requirements

The main functional requirement of this application is that it allows to anonymize the faces of the possible subjects appearing in the images taken in the field in real time, so that it is possible to send the video stream directly anonymized if required to avoid problems with GDPR legislation.

A second functional requirement is to add a run mode in which the anonymized output images maintain as realistic an appearance as possible. This option would be used in case of needing to anonymize images that are going to be part of a dataset, to avoid that the dataset contains bias due to the introduction of artifacts in the anonymized faces.

5.2.2.4 Data models

This application does not store data in any format, it takes a video stream as input and provides one or more video streams as output, in mp4/avi format, or individual frames, in jpg/png.

5.2.2.5 Application Programming Interfaces (APIs)

The definition of a final interface to interact with the tool is pending the implementation of the tool as an inference service in the FlexiGroBots AI platform. Currently, the software is executed as a python script inside a docker container.

5.2.2.6 Graphical User Interfaces (GUIs)

There is no GUI beyond the display of the output videos themselves.

5.2.2.7 Installation

To install this application and be able to run it, it is only necessary to have the required permissions in the repository of the project to download the app docker image, as following:

```
docker pull ghcr.io/flexigrobots-h2020/anonymization-tool
```

It is possible to check the possible input parameters of the application script through the following command:

```
docker run -it anonymization-tool --help
```

Once you have checked that the image is running properly and the possible input parameters are known, delete the previous container and run it again with the desired parameters.

5.2.2.8 Prototype availability within FlexiGroBots

The prototype of this application is available in the project's GitHub repository by downloading and executing the docker image named *anonymization-tool*.

Access to the code repository is private, only members of the repository are allowed. People linked to the project who are not members of the repository will be able to access on demand. The link to the private repository is: https://github.com/FlexiGroBots-H2020/anonymization-tool.

5.2.2.9 Release planning

User story ID	User story	Priority	Story points
CAS_AT_US_01	Evaluate and test in field conditions (Pilots 1, 2 and 3)	High	3
CAS_AT_US_02	Upload the application as AI platform inference service and prepare deployment for required devices	Medium	3

Table 10 User stories for anonymization

5.2.3 Automatic dataset generation

The Automatic dataset generation common application is introduced for the first time in this document, as it was not presented in the previous deliverable, D3.1. The need to develop this tool has arisen due to the lack of availability labelled images to train AI computer vision models in agricultural context.

As it is well known by people involved in the field of artificial intelligence, in general, and more specifically in the field of computer vision, the lack of labelled datasets is always one of the main hurdles to overcome when approaching a project.

FlexiGroBots is no exception, there is a need for very specific datasets for certain tasks, and although it is sometimes possible to find public datasets or manually label data collected in pilots, the reality is that it is always a tedious and complicated job. In the worst cases, it becomes impossible to find or produce enough data to be able to train a detection model, as was the case for the use case of tractor detection from aerial image, introduced in section 5.1.4.

The solution proposed with this common application allows to automatically generate tagged datasets of the desired concept. The pipeline developed is reflected in Figure 31.

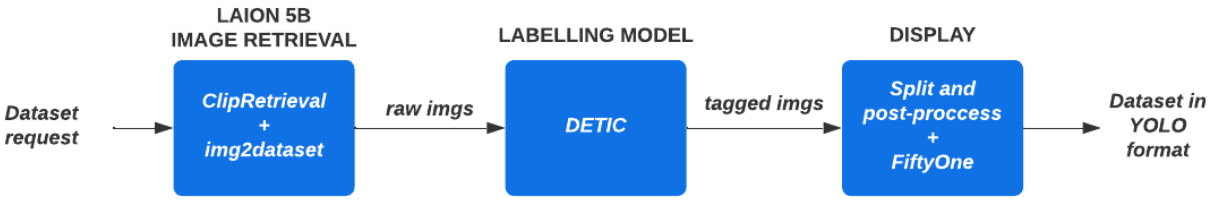

Figure 31 ADGT pipeline description

The defined pipeline includes 3 steps and implements 4 open-source repositories along with a multitude of proprietary developments. The resulting tool takes the field of automatic image labelling beyond the current state of the art.

The first step of the pipeline is the collection of unlabelled raw images, and for this purpose two different options have been developed. The first, the most direct and simple, is to indicate the path to a folder containing raw images or videos of the concept to be collected in the resulting dataset. The second is to use the open-source Clip-retrieval tool, [47], to search for images of the desired concept in the public LAION5b data lake [48]. The search can be performed by providing example images or by describing the desired scene. Figure 32 reflects these two options. Once similar images have been retrieved, the img2dataset [49] software is included to download the appropriate images from the source.

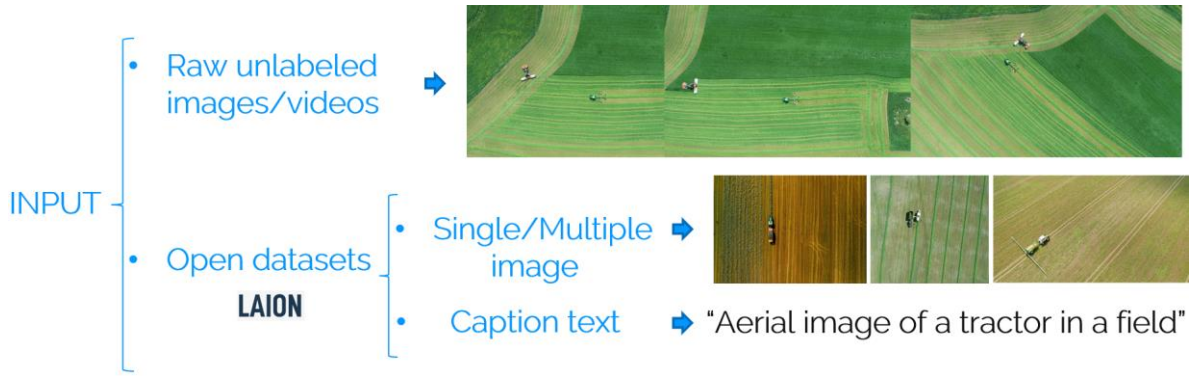

Figure 32. Dataset request input options for the first step: raw data collection

The second step of the defined pipeline is the automatic labelling of the images obtained in the previous step. Detic [34], a detection and segmentation model already introduced in one of the previous applications, 5.1.2, is implemented to carry out this labelling. This model is ideal for this task, since as explained in the referenced point, Detic is pre-trained in more than 21.000 different classes and is also capable of zero-shot classification for concepts not included among those classes. A sample of images from the LAION5b data lake tagged using Detic is shown in Figure 33.

Figure 33 Images automatically tagged with Detic [34], extracted from the LAION-5b [48] data lake

The third and last step is the visualization of the generated dataset, in order to check whether the result meets the quality required to train a detection model. For this purpose, the functionalities offered by the open-source FiftyOne [50] library are used, which, as shown in Figure 34, allows visualizing the results in a browser tab and interacting with the classes and datasets of the generated dataset.

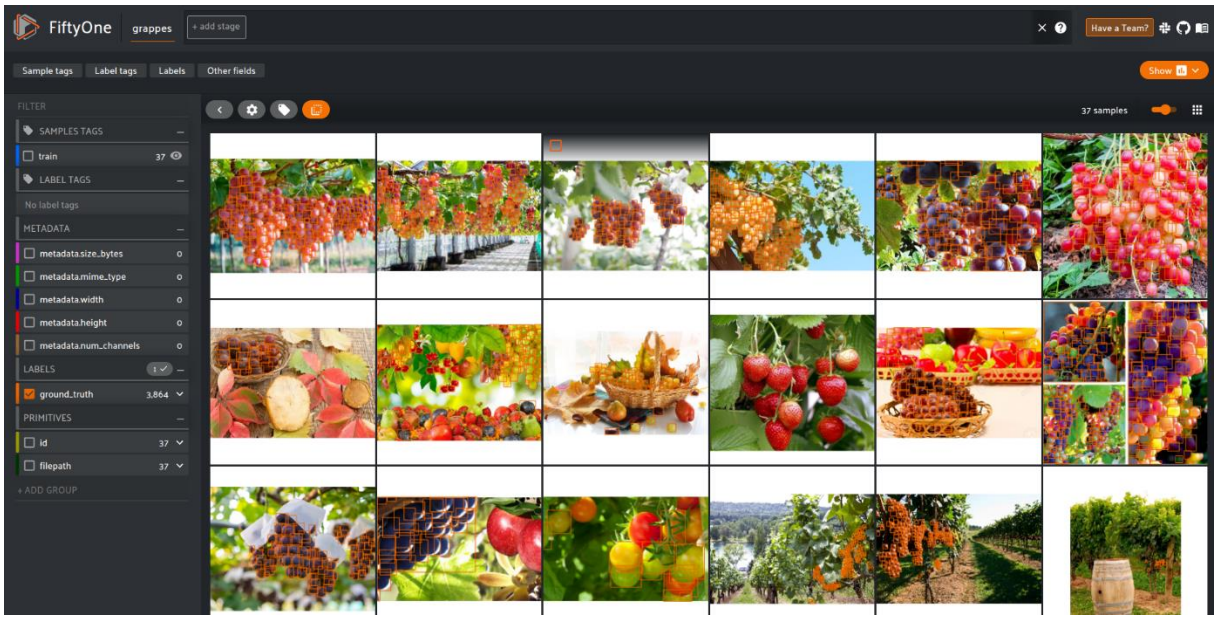

Figure 34. FiftyOne GUI for generated dataset visualization

As an example of the real use of this application, more information about the dataset created to complete tractor detection task can be found in Annex A.

The added value of this application is remarkable, introducing a new source of data to be considered when dealing with the complex process of training neural models for detection and segmentation tasks, since in addition to the bounding boxes of the detected objects, the segmentation masks are also extracted from them.

5.2.3.1 Implemented functionalities

The current implementation of the tool offers the following functionalities:

- Generation of labelled datasets from raw images and videos collected in the pilots.
- Image retrieval and labelled datasets generation based on similar unlabelled image examples.
- Image retrieval and labelled datasets generation based on a textual description of the desired concept.
- Online visualization of the generated dataset.
- Output format of the dataset ready to be used as input to train detection models, YOLO format.

5.2.3.2 Technical requirements

It is necessary to have sufficient disk space to be able to download and run the docker image, 15 GB, also enough GPU memory to host the labelling model, it requires 8 GB. It is highly recommended to have a Nvidia GPU to speed up the auto-labelling process of the data. This tool has been implemented in a Nvidia GeForce GTX 1070 with CUDA 11.3 and python3.8.

5.2.3.3 Functional requirements

Unlike the other applications, the functional requirements of this tool are not directly defined by the needs expressed by the agricultural stakeholders. In this case, the functional requirements are derived from the internal need for more labelled images to train AI models. Therefore, the main functional requirement is that the tool allows to obtain annotated images of concepts related to the agricultural context. The tool must be broad spectrum, i.e., be able to generate datasets of a wide number of different concepts within the project context, as well as fruits or vehicles. In addition, the tool should be easy to use and allow a user-friendly visualization of the results.

5.2.3.4 Data models

This application generates a structure of folders containing the data resulting from each new dataset request. On the one hand, unlabelled collected images are stored, and on the other

hand, images and annotation files are stored and divided into training, evaluation, and test sets. The folder architecture is shown in Figure 35.

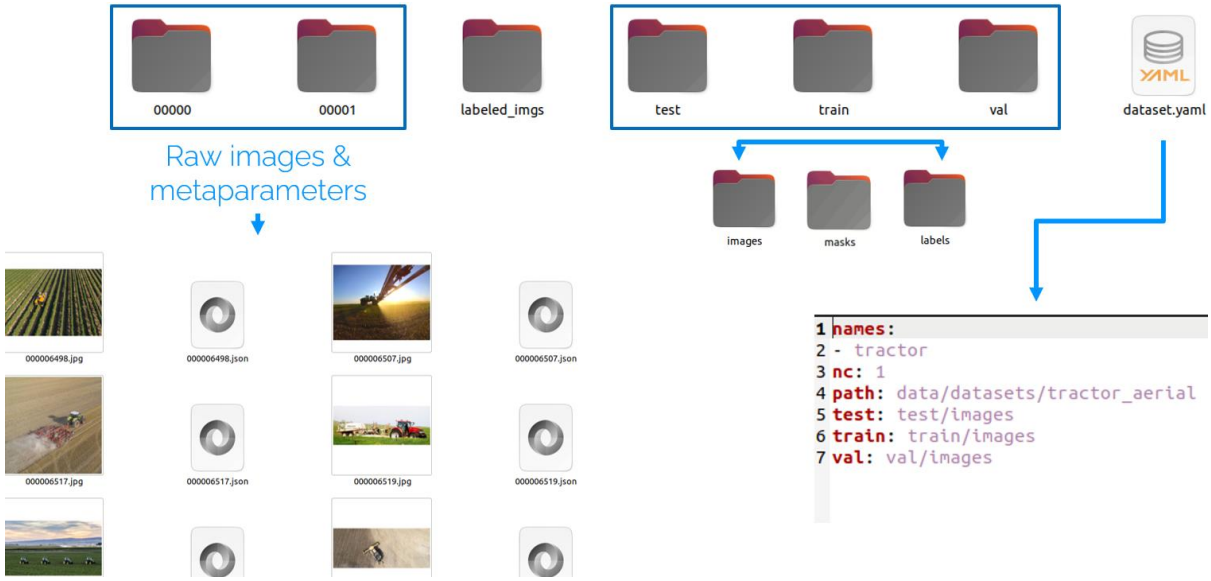

Figure 35 DGT output data architecture

5.2.3.5 Application Programming Interfaces (APIs)

The definition of a final interface to interact with the tool is pending the implementation of the tool as an inference service in the FlexiGroBots AI platform. Currently, the software is executed as a python script inside a docker container.

5.2.3.6 Graphical User Interfaces (GUIs)

In addition to the dataset files that remain in the folder from which the application is launched, a browser tab opens with a GUI that allows you to interact with some meta-parameters of the generated dataset. This interface is reflected in Figure 34.

5.2.3.7 Installation

To install this application and be able to run it, it is only necessary to have the required permissions in the repository of the project to download the app docker image, as following:

```
docker pull ghcr.io/flexigrobots-h2020/dataset-generation-tool
```

It is possible to check the possible input parameters of the application script through the following command:

```
docker run -it dataset-generation-tool --help
```

Once you have checked that the image is running properly and the possible input parameters are known, delete the previous container and run it again with the desired parameters.

5.2.3.8 Prototype availability within FlexiGroBots

The prototype of this application is available in the project's GitHub repository by downloading and executing the docker image named *dataset-generation-tool*.

Access to the code repository is private, only members of the repository are allowed. People linked to the project who are not members of the repository will be able to access on demand. The link to the private repository is: https://github.com/FlexiGroBots-H2020/dataset-generation-tool.

5.2.3.9 Release planning

User story ID	User story	Priority	Story points
CAS_DGT_US_01	Guidance on dataset generation (Pilots 1, 2 and 3)	High	3
CAS_DGT_US_02	Upload the application as AI platform inference service and prepare deployment for required devices	Medium	3

Table 11 User stories for automatic dataset generation

5.3 Generalization

As introduced in D3.1, this third group of applications is oriented to generalize the tools developed within the context of each pilot, so that they are applicable to general use cases within the field of agriculture, or at least can be used as a starting point to develop new solutions. Also included in this section are tools that solve generic problems when deploying solutions for the tasks of (i) disease detection, (ii) pest detection, and (iii) weed detection.

The fact that these applications are intended to generalize the applications developed in the pilots means that their implementation is subject to these applications being in a final phase of their development; therefore, it is not possible to count on very advanced common applications at this point in the project.

5.3.1 Disease detection

The definition of the task and the approach reflected in deliverable D3.1 remain in force. Briefly recapitulating, a model for detection of botrytis disease in grapes continues to be developed within the context of Pilot 1, and the idea is to generalize this architecture and the defined pipeline, to be applicable to other fruits and diseases once the implementation is completed.

For this reason, efforts up to this point have been focused on offering general functionalities that can be used both by the pilots of this project and other possible solutions in the field of agriculture when detecting diseases affecting crops or fruits.

The tool implemented as a result of task T3.4 allows detecting and segmenting fruits as a previous step to the detection of diseases in them. In this way, using the segmented image with the extracted background as input to the disease classification model, it's possible to minimize the noise introduced to the model and facilitates its training by reducing the complexity of the entry space. Several examples with blueberries and grapes can be seen in Figure 36 and Figure 37 respectively.

The main component of the tool is Detic [34], which was introduced in previous points, 5.1.2, is a DETR-style detector able to make inference even with unspecified classes during the training phase thanks to the introduction of CLIP. For more details visit the referenced section and the included citations.

The added value of this solution is to reduce the complexity of the problem before facing the disease detection stage, which is always desirable to maximize the chances of obtaining good results.

Figure 36 Blueberries detection and segmentation with FlexiGroBots' common app

Figure 37 Grapes detection and segmentation with FlexiGroBots' common app

5.3.1.1 Implemented functionalities

The current implementation of the tool offers the following functionalities:

- Multiple fruits detection and segmentation.
- Background extraction.
- Fruit counting.

5.3.1.2 Technical requirements

It is necessary to have sufficient disk space to be able to download and run the docker image, 15 GB, also enough ram to host Detic model, 6 GB. It is highly recommended to have a Nvidia GPU to speed up the detection and segmentation process of the data. This tool has been implemented in a Nvidia GeForce GTX 1070 with CUDA 11.3 and python3.8. The fps ratio obtained in under this scenario its approximately of 1, highly depending on the number of objects detected in the image.

5.3.1.3 Functional requirements

The nominal operation of this tool should allow to introduce image or video as input and to obtain the processed result with the detections and the classification of the state of the fruit in a relatively short period of time, without the need to reach real time.

In any case, the result of the analysis carried out should be recorded in one way or another, either in the same output images or in a textual message, but in case of detecting diseases in the fruit it would be desirable to send an alert to the MCC with the position of the event to carry out corrective actions.

5.3.1.4 Data models

This application does not store data in any format, it takes a video stream as input and provides one or more video streams as output, in mp4/avi format, or individual frames, in jpg/png.

5.3.1.5 Application Programming Interfaces (APIs)

The definition of a final interface to interact with the tool is pending the implementation of the tool as an inference service in the FlexiGroBots AI platform. Currently, the software is executed as a python script inside a docker container.

5.3.1.6 Graphical User Interfaces (GUIs)

There is no GUI beyond the display of the output images/videos themselves.

5.3.1.7 Installation

To install this application and be able to run it, it is only necessary to have the required permissions in the repository of the project to download the app docker image, as following:

```
docker pull ghcr.io/flexigrobots-h2020/disease-detection-tool
```

It is possible to check the possible input parameters of the application script through the following command:

```
docker run -it disease-detection-tool --help
```

Once you have checked that the image is running properly and the possible input parameters are known, delete the previous container and run it again with the desired parameters.

5.3.1.8 Prototype availability within FlexiGroBots

The prototype of this application is available in the project's GitHub repository by downloading and executing the docker image named *disease-detection-tool*.

Access to the code repository is private, only members of the repository are allowed. People linked to the project who are not members of the repository will be able to access on demand. The link to the private repository is: https://github.com/FlexiGroBots-H2020/fruit-disease-detection-tool.

5.3.1.9 Release planning

User story ID	User story	Priority	Story points
CAS_GDIS_US_01	Check the software module and requirements for Pilot 1's *Botrytis* detection	High	5
CAS_GDIS_US_02	Build domain adaptation and/or transfer learning module for allowing re-training of the Botrytis model	High	13
CAS_GDIS_US_03	Evaluate and test in field conditions with data from Pilot 3	Medium	3
CAS_GDIS_US_04	Upload the application to the AI platform and prepare deployment for required devices	Medium	3

Table 12 User stories for disease detection

5.3.2 Pest detection

Analogous to the point 5.3.1, the description of this tool was included in document D3.1, and its progress is subject to that of the tool developed in the context of Pilot 2 to detect the pests of "meligethes aeneuss" in rapeseed plants.

After having studied and tested different alternatives to face the problem, it has been concluded together with the stakeholders of Pilot 2, that the most viable approach is to make an estimation of the affection of the crops according to the number of insects detected in yellow sticky traps placed among the crops.

One of the favorable points to reach this conclusion is that these traps can be prepared with different pheromones to attract different species of insects, thus being a generalizable solution for other pests and crops.

Another reason for this decision is that after taking a set of hundreds of images in the field, even in cases where the insects are not blown away by the air generated by the UAV taking the image, it is very difficult to locate the insects on the crops, making it impossible to obtain a dataset with enough samples to train a detector. Figure 38 shows how complex the problem would be posed in this way.

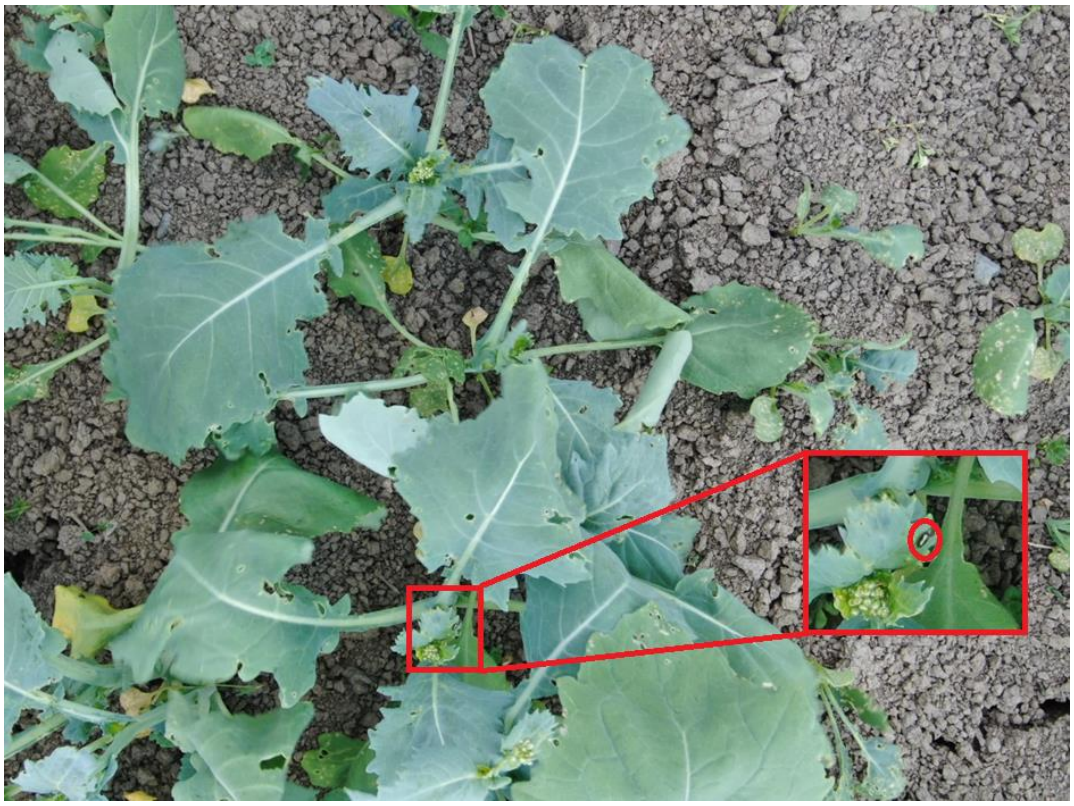

Figure 38 Pilot 2 raw image example for “meligethes aeneuss” detection

Therefore, the application proposed at this point of the project implements Detic [34] as model to segment the traps and detect the insects, but in future iterations it is intended to include models of concrete use to improve the performance obtained in the detection and classification of trapped insects. This requires specific datasets for the insect species of interest. Figure 39 shows some of the results that can be obtained using the implemented tool.

Figure 39 Trap segmentation and insect detection with FlexiGroBots' common app

A visual description of the pipeline defined at this point in the project can be seen in Figure 40.

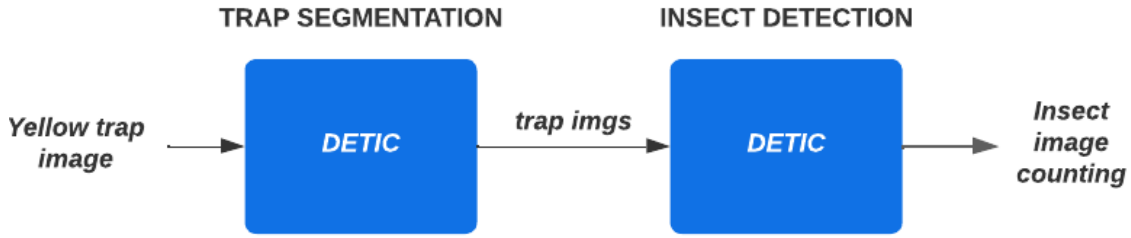

Figure 40 Pest detection pipeline description

The added value of this solution is that based on field experimentation, conclusions have been drawn that lead to discarding other alternatives and implementing a two-step solution, yellow trap segmentation and insect detection/counting, which is generally applicable to the generic pest detection in crops problem.

5.3.2.1 Implemented functionalities

The current implementation of the tool offers the following functionalities:

- Multiple types of insect detection and segmentation. Also, generic insect class.
- Yellow trap segmentation and background extraction.
- Insect counting.

5.3.2.2 Technical requirements

It is necessary to have sufficient disk space to be able to download and run the docker image, 15 GB, also enough ram to host Detic model, 6 GB. It is highly recommended to have a Nvidia GPU to speed up the detection and segmentation process of the data. This tool has been implemented in a Nvidia GeForce GTX 1070 with CUDA 11.3 and python3.8. The fps ratio obtained in under this scenario its approximately of 1, highly depending on the number of objects detected in the image.

5.3.2.3 Functional requirements

The nominal operation of this tool should allow to introduce image or video as input and to obtain the processed result with the detections and the classification of the state of the pest presence in a relatively short period of time, without the need to reach real time.

In any case, the result of the analysis carried out should be recorded in one way or another, either in the same output images or in a textual message, but in case of detecting pest in the crop it would be desirable to send an alert to the MCC with the position of the event to carry out corrective actions.

5.3.2.4 Data models

This application does not store data in any format, it takes a video stream as input and provides one or more video streams as output, in mp4/avi format, or individual frames, in jpg/png.

5.3.2.5 Application Programming Interfaces (APIs)

The definition of a final interface to interact with the tool is pending the implementation of the tool as an inference service in the FlexiGroBots AI platform. Currently, the software is executed as a python script inside a docker container.

5.3.2.6 Graphical User Interfaces (GUIs)

There is no GUI beyond the display of the output images/videos themselves.

5.3.2.7 Installation

To install this application and be able to run it, it is only necessary to have the required permissions in the repository of the project to download the app docker image, as following:

```
docker pull ghcr.io/flexigrobots-h2020/pest-detection-tool
```

It is possible to check the possible input parameters of the application script through the following command:

```
docker run -it pest-detection-tool --help
```

Once you have checked that the image is running properly and the possible input parameters are known, delete the previous container and run it again with the desired parameters.

5.3.2.8 Prototype availability within FlexiGroBots

The prototype of this application is available in the project's GitHub repository by downloading and executing the docker image named *pest-detection-tool*.

Access to the code repository is private, only members of the repository are allowed. People linked to the project who are not members of the repository will be able to access on demand. The link to the private repository is: https://github.com/FlexiGroBots-H2020/pest-detection-tool.

5.3.2.9 Release planning

User story ID	User story	Priority	Story points
CAS_GINS_US_03	Evaluate and test in field conditions with data from Pilot 3	Medium	5
CAS_GINS_US_04	Upload the application to the AI platform and prepare deployment for required devices	Medium	3
CAS_GINS_US_05	Obtain dataset of insects trapped in yellow traps	High	5
CAS_GINS_US_06	Implement a generic grading solution for various types of insects	High	13

Table 13 User stories for pest detection

5.3.3 Weed detection

Analogous to points 5.3.1 and 5.3.2, the description of this tool was included in document D3.1, and its progress is subject to that of the tool developed in the context of Pilot 3 to detect weeds in blueberry farms.

A similar approach to the other applications of the generalization cluster has been tried to be implemented in this use case, using Detic as detector and segmentator, but the model does not offer good results for this application. Figure 41 shows a case processed with the tool, in which it has not been possible to differentiate correctly between crop and weeds, obtaining as a result a segmentation that loses elements and misclassified it. The dataset [51] used to carry out this test presents images like the one in the figure, with ground truth segmentation masks and bounding boxes.

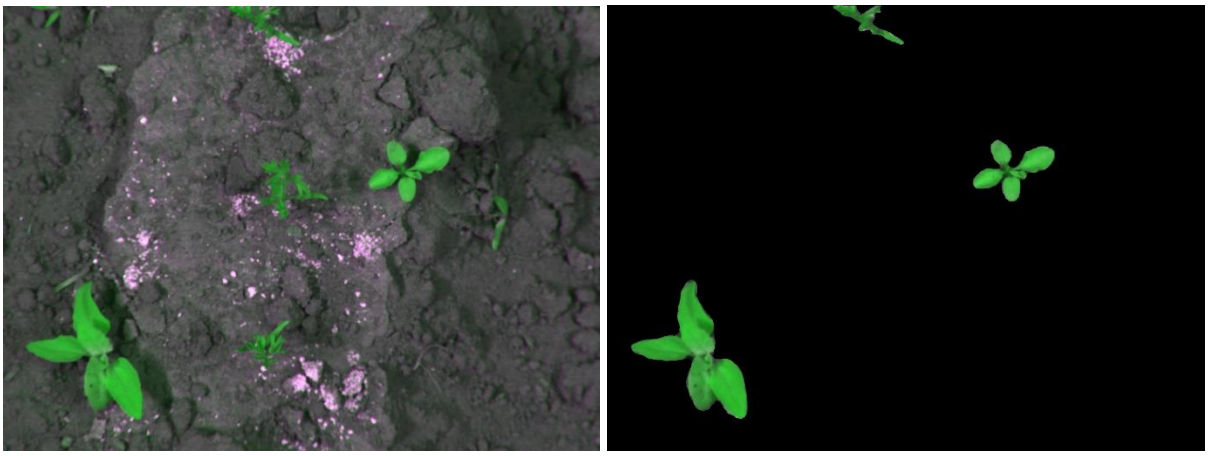

Figure 41 Weed/Crop segmentation with FlexiGroBots' common app on [51]

Given the poor results obtained, the development of a final solution for this application is pending for the last period of the project, and we will try to provide a generic solution based on the proposal deployed in Pilot 3.

5.3.3.1 Implemented functionalities

No functionalities implemented yet, waiting for the output of task T6.2.

5.3.3.2 Technical requirements

The technical requirements will be better defined once a prototype of the solution is reached, for the moment it can be assumed that an Nvidia GPU will be highly recommended to accelerate the execution of the models.

5.3.3.3 Functional requirements

The nominal operation of this tool should allow to introduce image or video as input and to obtain the processed result with the detections and the classification of the weed/crop presence in a relatively short period of time, without the need to reach real time.

In any case, the result of the analysis carried out should be recorded in one way or another, either in the same output images or in a textual message, but in case of detecting weeds in the crop it would be desirable to send an alert to the MCC, with the position of the event to carry out corrective actions.

5.3.3.4 Data models

Predictably, this application will not store data in any format, but takes a video or individual images as input and provides processed video or images as output.

5.3.3.5 Application Programming Interfaces (APIs)

The definition of an interface to interact with the tool is pending on the develop of the tool and its implementation as an inference service in the FlexiGroBots AI platform.

5.3.3.6 Graphical User Interfaces (GUIs)

There will foreseeably be no graphical user interface beyond the display of the output images/videos themselves.

5.3.3.7 Installation

Not yet defined, pending the development of the tool.

5.3.3.8 Prototype availability within FlexiGroBots

There is no prototype ready for general use by FlexiGroBots project stakeholders yet.

5.3.3.9 Release planning

User story ID	User story	Priority	Story points
CAS_GWEED_US_01	Check the software module and requirements for Pilot 3's weed detection model	High	5
CAS_GWEED_US_02	Extend the weed detection module with customization features so it can be reused in other domains	High	13
CAS_GWEED_US_03	Evaluate and test in field conditions with data from Pilot 1	Medium	3
CAS_GWEED_US_04	Upload the application to the AI platform and prepare deployment for required devices	Medium	3

Table 14 User stories for weed detection

6 Mission Control Centre

Mission Control Centre (MCC) is responsible for multi-robot operations. During the second project year some revisions to original MCC concept were made based on observations from project's pilots. The main topics were:

- The pilots' robots are autonomous devices that need little control from the user during the execution of tasks. They do not need additional low-level control. The robot systems consist of the full stack of components. They do have their own specific task planning tools and operation control systems. Therefore, a loosely coupled fleet control approach was chosen. It means that only high-level commands such as pause, resume, change task are needed and that the planning of individual robot's task is done by robot specific tools.
- The robots' control systems are more advanced than assumed meaning that they have file systems and that their tasks are described as files. The firmware operations and task dispatchers can be replaced with more normal file operations.
- The use cases were consisting of sequential robot or robot fleet tasks completed with analysis and design tasks. It is more like a workflow that must be completed. Field operations are either producing data for next steps or dependent on the data from previous steps. In some cases, these steps can involve external services or decisions of the farmers or the robot operators. In order to specify, design, and manage such system, the mission control centre has to manage these other types of tasks and data transfers between them.

New definitions were created:

- Definition of a mission workflow: *Mission workflow* is a collection of actions to be done for achieving farm level objectives. Mission workflow may have several *phases* that consist of single robot or fleet *tasks*, use of AI or data services, or user tasks for decision making or planning of next steps.
- Definition of a fleet task: Fleet task is an activity at the field done simultaneously by a set of heterogenous, collaborating robots.
- External service: External service is a service that processes data given as an input to the mission or created by a phase of the mission, and that produces new data to be used in a mission.
- User tasks: User task is a task performed by a mission stakeholder for making decisions, planning new actions, or preparing data for next phases.

The architecture of MCC (Figure 42) was redesigned based on the new approach. The main idea is to separate the control and coordination of actions into three levels.

- The lowest level is the field robot system that includes the robot control that is done using the robot or robot swarm controller[11]. They are specific to robots and robots' manufactures and as our approach is to focus on fleet level control it is out of scope of the MCC. The robot or robot swarm controller must, however, be able to communicate with MCC and robot tasks must be aware of the fleet in some cases.
- The middle level is the fleet management level that coordinates the operation of the fleet.
- The upmost level is the mission management level that has responsibility for the mission workflow planning, coordination, and communication with external farm management system.

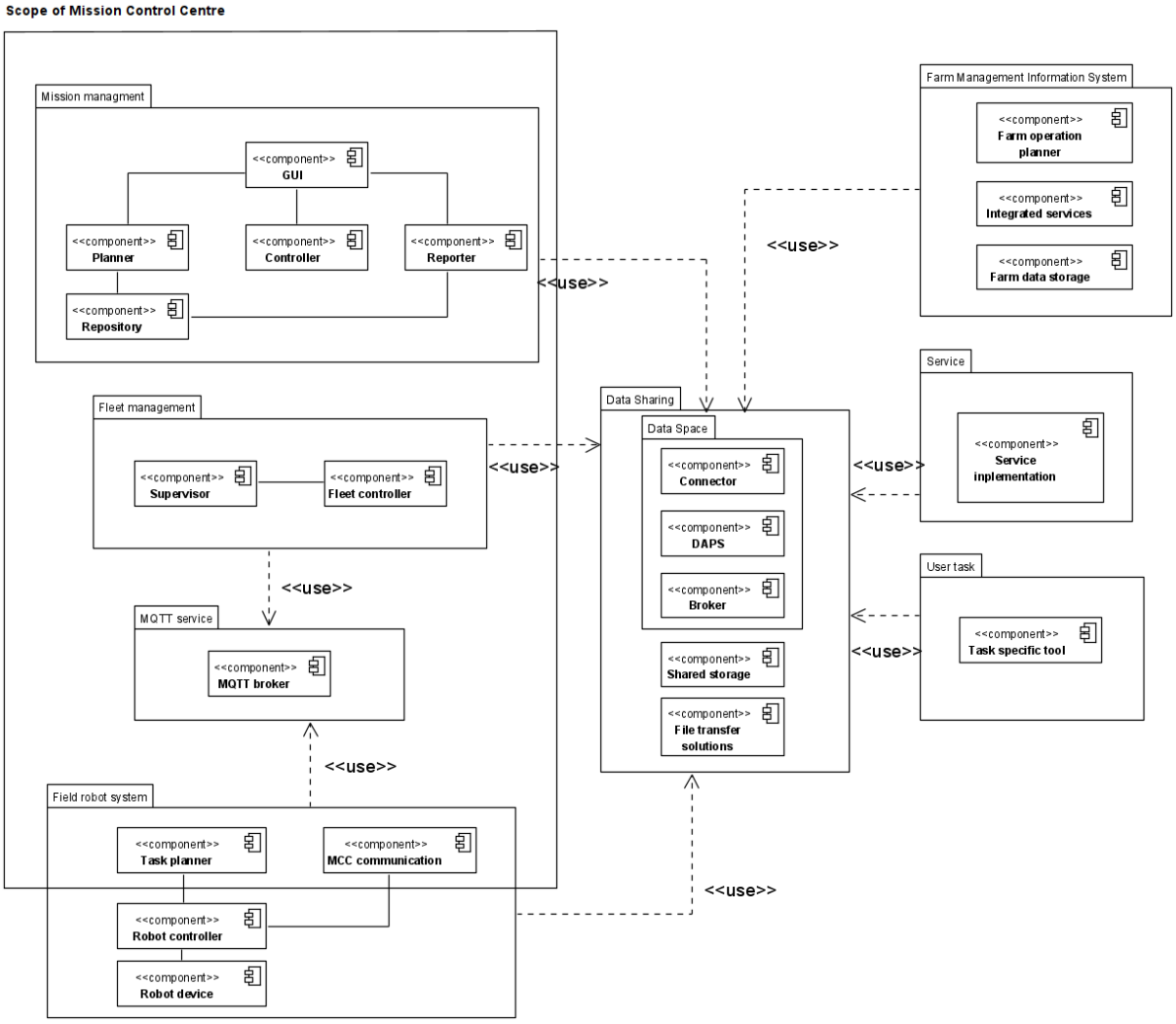

Figure 42. Architecture diagram of Mission Control Centre

[11] Robot swarm is a collection of robots controlled by a single controller. In the MCC context the robot swarm is considered equal to robot as all the communication takes place through the controller.

The control and coordination techniques follow the three levels. The robot system level everything is based on robot specific techniques not covered here. In the fleet and mission level we use publish-subscribe approaches.

At the fleet level communication is based on MQTT protocol and command and status payloads specified in the FlexiGroBots projects. Status and command messages are based on Ultralight protocol. Every actor in the fleet operation publishes their messages in MQTT broker and subscribe to messages they need.

At the mission workflow level the communication is based in IDSA data space, shared data storages or other data transfer protocol depending on the deployment scenario. The reference implementation will support the data space approach. Different entities in each phase of the mission publish the data for selected receivers and subscribe to data they need from other entities.

Mission workflow description (*Mission File*) is the key element in the system. It is a JSON description of the mission containing the farm data, communication system configurations, data elements, phases with robot fleets, services, and user tasks. Mission File is described in more detail in section 6.2.3.

Components and deployment

The developed architecture supports various deployment scenarios. Different use cases have different requirements and the MCC system must fulfil them. For example, the location of the farm may be such that the Internet connections do not allow even near real-time control features and the fleet controller, fleet supervisor, MQTT messaging, and robot control systems must be deployed into local computing environment. Another possible factor may be the ownership of robots and devices. The architecture supports subcontracting-based robot fleet tasks and that may require that fleet control and supervision is done using robot operator's systems. Description of the components and their deployment options in different types of mission flows is described in Table 15 and deployment example in Figure 43.

Component	Description	Deployment scenarios
Mission workflow planner	A tool for creating, editing, and managing mission file	A service in a cloud. Hosted by FlexiGroBots platform or other service provider
Mission workflow controller	A too for managing the mission workflow execution. Controls the acceptance and launch of phases and manages the data transfers	A service in a cloud. Hosted by FlexiGroBots platform or other service provider
Mission reporter	A tool for creating a mission report for a farm management information system	A service in a cloud. Hosted by FlexiGroBots platform or other service provider

Component	Description	Deployment scenarios
Mission repository	Data storage for mission files and report data	A service in a cloud. Hosted by FlexiGroBots platform or other service provider
Mission management GUI	REACT interface to manage mission manager tools	Web application.
Fleet controller	A tool for controlling the fleet of robots. Control operations are starting, stopping, and changing robot tasks, and pausing and resuming the active task	Stand-alone application with UI, web application and service, integrated service to farm management systems, or extension to QGroundControl software.
Fleet supervisor	A tool for visualising the robot fleet status	Stand-alone application with UI, web application and service, integrated service to farm management systems, or extension to QGroundControl software.
MQTT broker	Service providing MQTT message publication and subscriptions. Existing component	A service in server or local PC.
MCC communication	A component providing MCC messaging capabilities for robot systems and Mission File interface	Tool is integrated to fleet controller, fleet supervisor, and robot control system.
Robot task planner	Robot task planning tools manage a robot or a robot swarm considering each robot type characteristics, and they are developed in pilots. The resulting plans for each robot must be linked to mission description file. The UGV robot route planning tool is developed as a part of MCC (at the field robot system level) as route planning is a general component needed in all UGVs.	Robot route planner is integrated to robot task planner. Robot task planner can be a stand-alone application of web service. Run-time replanning of robot's route is considered, but possible implementations are done in pilots and not necessarily using the fleet route planner.

Table 15 Main parts of MCC and their deployment options

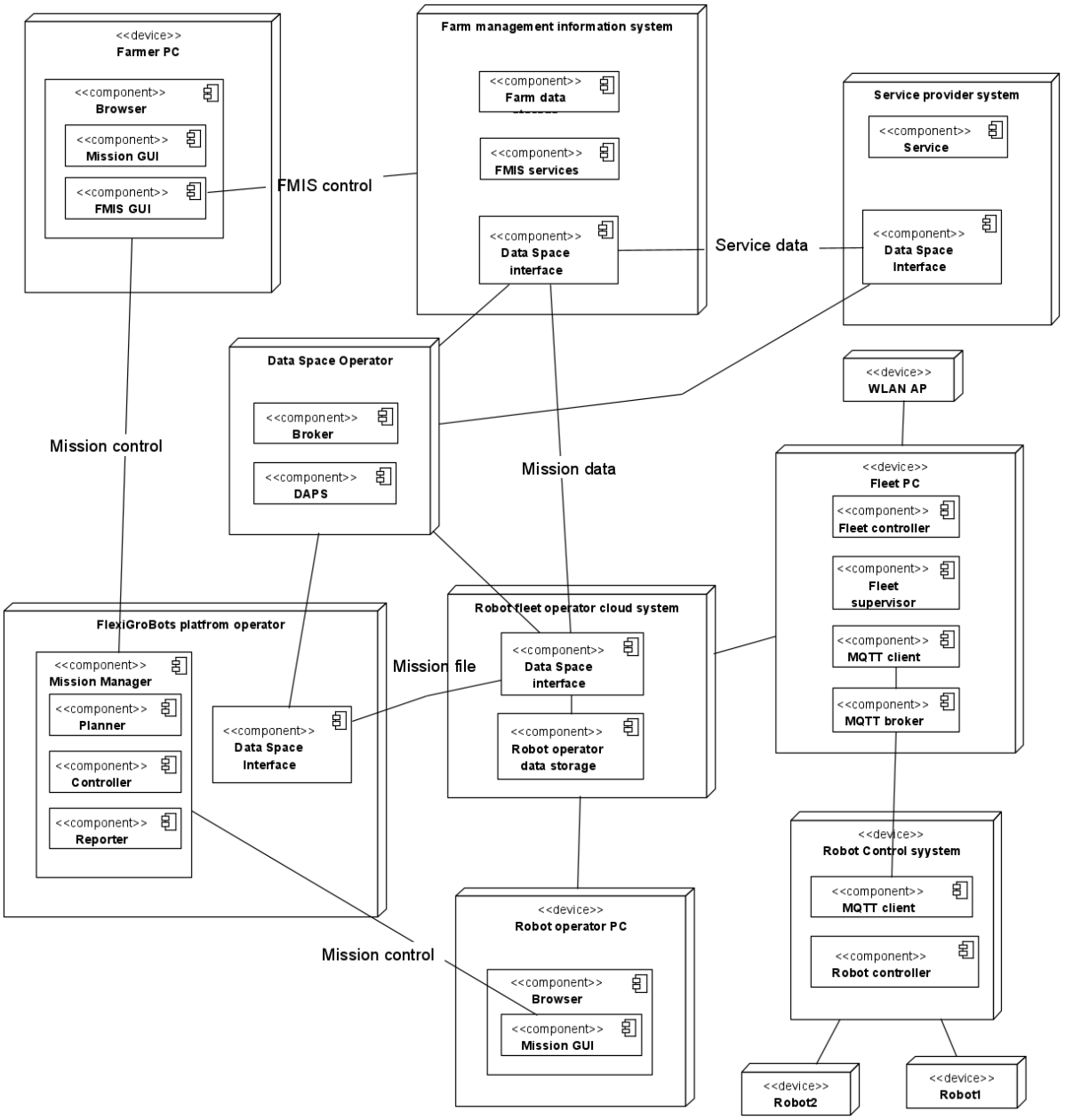

Figure 43. MCC deployment example

6.1 Implemented functionalities

The implemented functionalities are described in Table 16 (functionalities described in D2.3) and Table 17 Table 16 (new functionalities for managing mission workflow and phases). In these tables we have described the components involved and overall status of implementations. It must be noted that some earlier user stories have currently slightly changed interpretations. As the implementation planning is currently focusing on development of architecture components the status is component implementation is also given in Table 18.

User story ID	User story	Components involved	Comment
MCC_US_01	Create a simulation environment for development purposes	Fleet controller, MCC communications, MQTT broker, robot simulators	A python fleet controller and robot simulator with MQTT communication has been developed. There are also simulators for drones and autonomous tractor.
MCC_US_02	Create controllers with ground robots in pilot 1 to send missions	Fleet controller data space (for creation of fleet management configuration)	Ground robot controllers and robot task plan management are pilot-specific implementations. In pilot 1 robot specific controllers are used and they are on board the robots.
MCC_US_03	Create controllers with ground robots in pilot 2 to send missions	Fleet controller, data space (for creation of fleet management configuration)	Ground robot controllers and robot task plan management are pilot-specific implementations. In pilot2 ISOBUS task controller and EFDI communication, QGroundControl and MAVIC drone plans and Weeding robot specific controller are used.
MCC_US_04	Create controllers with ground robots in pilot 3 to send missions	Fleet controller, data space	Ground robot controllers and robot task plan management are pilot-specific implementations. Robot-specific controllers are used in pilot 3.
MCC_US_05	Create supervisors with ground robots in pilot 1	Fleet supervisor and controller, MCC communications, MQTT broker	PoC fleet controller has been developed. It implements MCC status and command messaging and fleet status visualisation interface.
MCC_US_06	Create supervisors with ground robots in pilot 2	Fleet supervisor, fleet controller, MCC communications, MQTT broker	
MCC_US_07	Create supervisors with ground robots in pilot 3	Fleet supervisor, fleet controller, MCC communications, MQTT broker	
MCC_US_08	Implement planner to support missions with multiple robots	Mission planner, robot task planner	Initial mission planner exists.

User story ID	User story	Components involved	Comment
MCC_US_09	Implement events processing and alarms notification	Fleet controller	Not implemented yet.
MCC_US_10	Implement Data Space connector	Data Space Connector, broker, and DAPS	IDSA data space with connectors, broker and DAPS has been developed and tested. There are two deployments (in pilot 2 and FlexiGroBots platform)
MCC_US_11	Implement MQTT communication extension for QGC	Fleet supervisor and controller	MQTT extension to QGC has been developed.

Table 16 Mapping of functional requirements from D2.3 to MCC components

User story ID	User story	Components involved	Description
MCC_US_12	Create a model for complete mission workflow and its phases	Mission workflow planner	Mission file specification as a JSON schema has been developed.
MCC_US_13	Support multi-partner execution of mission	Communication between all the components. Data space.	A model on how different partners communicate and exchange confidential data across data space has been created.
MCC_US_14	It should be possible to share data created in robot missions with AI development platform	Data space connectors, mission manager	A model on how the data sharing and data management during a mission has been developed. MCC architecture and components support the model.

Table 17 Mapping of new user stories to MCC components

Component	Implementation status	Comments
Mission workflow planner	New component JSON-schema for mission file has been designed and tested using REACT test environment	Planner will be based on existing JSON server and a REACT user interface. Interfacing to data sources related to robot operators are under work.
Mission workflow controller	New component. Specification exists	-
Mission reporter	New component. Specification exists	-

Component	Implementation status	Comments
Mission repository	Repository is standard server data storage system	-
Mission management GUI	New component. Initial specification exists	GUI will be based on REACT technology.
Data space interface	Extension to IDSA connect. Interface to data space from MCC have been developed based on IDSA components. Infrastructure has been tested	Interface is based on existing components, but it has need additions related to security, notifications, and to use guidelines.
Fleet controller	Extension (new features) to QGC. Python based fleet controller with MQTT and Mission File interfaces has been developed	Current components implement the MCC messaging capabilities, visualisation, and control options. Fleet controller and supervisor will be integrated both to QGroundControl and Farm Management System as a web application.
Fleet supervisor	Extension (new data to be displayed) to QGC Python based fleet supervisor and initial QGroundControl extension have been developed. Both have MQTT interfaces	
MQTT broker	MQTT is an existing component	Current deployment is using Eclipse Mosquito MQTT broker. Different MQTT brokers can be used in the execution of fleet operations.
MCC communication	New component. Reference implementation of MQTT and MissionFile interface have been done	Implementation is based on Python3 JSON and paho.mqtt.client libraries.
Robot task planner	New components. Matlab version is available and comes from previous projects	Work is underway for having a communication based in IDSA data space to get maps generated by the UAVs and to share the plans generated to other parts of the MCC architecture. In addition, modifications are being made to properly adapt the planner behaviour to the new scenarios.

Table 18 Implementation status of main MCC components in FlexiGroBots platform v1

6.2 Requirements

6.2.1 Technical requirements

The Mission Control Centre itself is technically relatively simple software. The technical requirements depend heavily on the intended use and deployment scenarios, so it is difficult to give exact general numbers. The numbers and estimations here are based on the reference implementations developed in the project and project's use cases. These numbers should be considered initial due to current maturity level of implementations.

- The mission management will be a web service deployed as Docker container. The technical requirements for execution environment depend heavily on the number of users. However, the computation load caused by individual user is small.
- The fleet management will be developed as an extension to QGroundControl (QGC) software. The additional load caused by the extensions will be negligible, so the initial requirements of QGC given in D3.1 will be sufficient.

As the data transfers will be done using the IDSA data space solution, the companies that want to be part of the business ecosystem, must have data space connectors. The companies must have at least a server connected to Internet with fixed IP address. The companies must also have certified IDSA connectors and identities. Technical requirements for IDSA connectivity platform basically the same as for Docker virtualisation:

- 64-bit kernel and CPU support for virtualization.
- KVM virtualization support. Follow the KVM virtualization support instructions to check if the KVM kernel modules are enabled and how to provide access to the KVM device.
- QEMU must be version 5.2 or newer. We recommend upgrading to the latest version.
- systemd init system.
- Gnome, KDE, or MATE Desktop environment.
- For many Linux distros, the Gnome environment does not support tray icons. To add support for tray icons, you need to install a Gnome extension. For example, AppIndicator.
- At least 4 GB of RAM.
- Enable configuring ID mapping in user namespaces, see File sharing.

The communication between fleet management and robot systems requires a MQTT broker and a computer hosting it. In practice almost any server or computer fulfils them. Even a Raspberry Pi can run a MQTT broker.

To run the robot task planner, it is necessary to have one of the latest Matlab 7.13 (R2011b) version installed.

The robot task planner has been successfully run-on Windows XP, Windows 7, Linux, and Mac OS systems. The only problems detected when running the code on these different operating systems were related to the save path of the data and in particular to the characters '/' and '/'. If you want to run the scheduler on a non-Windows system, you may need to tweak the data save functions to support the new system's paths.

If you are going to run the planner via interface or intend to develop part of it, you need to install a version of Matlab that includes the GUIDE utility (Matlab interface development tool).

If the planner is to be exported to C or C++, either directly to source code or to a static library, the Matlab Coder utility must be installed (Matlab versions higher than 7.13).

To include the code automatically generated by Matlab Coder in a C/C++ project, it is enough to have installed a C/C++ compiler supported by Matlab. These compilers can be consulted in the following link for Matlab version R2012a:

http://www.mathworks.es/support/compilers/R2012a/win32.html

Summary information on how the robot task planner works has been included in Annex B of this document.

6.2.2 Functional requirements

The functional requirements have evolved during the implementation design of the MCC. The original user stories were vehicles' provision, mission plan creation, mission execution, and mission supervision. The mission concept was extended to more holistic definition that is based on farmer's objectives in managing his crops. The main user stories were reformulated as mission planning, mission control, mission phase management (robot fleet, service, and user tasks), and mission reporting. These new user stories have been transformed to the requirements of the implementation components and the overall functionality of the system is built on top of two publish-subscribe systems, i.e., IDSA data space and MQTT. The next chapter will list the main functional requirements of each subsystem.

6.2.2.1 Mission workflow planner

The functional requirements of the mission workflow planner are:

- the creation of mission plan as a mission file in JSON format
- the mission file management (create, read, write, delete) operations using the mission repository
- the mission planner must be accessible by multiple operators across Internet
- mission planner must keep the plan confidential.

6.2.2.2 Mission workflow controller

The functional requirements of the mission workflow controller are:

- keeping record of the mission phase statuses
- activation of the phases (starting execution)
- initiation or execution of data transfers between the mission partners
- activation of the mission plan changes with the mission planner
- acceptance of the phase results
- ending of a phase
- ending of the mission and activation of mission reporting
- the mission controller is accessible by a mission owner (typically a farmer).

6.2.2.3 Mission reporter

The functional requirements of mission reporter are:

- collection and management of the mission data
- adding/editing of the mission description
- initiation of the data transfer to target farm management information system
- the mission reporter is accessible to mission owner.

6.2.2.4 Mission repository

Mission repository is a data repository accessible by mission management tools. Typical contents of mission repository are mission files, data collected and created by robots and device, and data created in missions' services.

Repository must be accessible by the Data Space connector.

6.2.2.5 Fleet supervisor

The functional requirements of fleet supervisor are:

- to be able to connect to MQTT broker and to subscribe to MQTT for MCC status messages
- to visualize the status and locations of the robots that participate to the fleet task.

6.2.2.6 Fleet controller

The functional requirements of fleet controller are:

- to be able to connect to MQTT broker and to send to MCC command messages
- to be able to send Pause, Resume, Start task, Stop task and Change task messages either to all or selected set of robots. In change status message the controller has to select the identifier of new task.

6.2.2.7 Robot task planner

The Matlab version of the robot task planner does not need to be installed. Simply copy all Matlab source code files to a working folder and access it from Matlab by making the Current Folder field point to that folder.

6.2.2.8 Description of how MCC is used in a complete use case

The operation flow of MCC and its phases are shown in Figure 44 and Figure 45. The main ideas in the mission process are that

1. the process completes during the execution
2. the mission file transform to mission report.

The complex farm operations that are the missions cannot be pre-planned as the complete execution flow may depend on the data that is created. For example, in the pest management the precision spraying cannot be designed in detail, before we know where the pests are. So, the mission can be planned only to the phase where we know all the input data. In fact, each use case has different needs, for example, in pilot 1 for Botrytis detection, it is possible to have pre-planned routes, based in the aerial scouting, for the ground robots performing the ground inspection and treatment tasks.

The mission produces data during the execution. The data can be related to the observations of the robot as in case of imaging survey of pests, or it can be data related to the execution of the task. Both data examples are valuable. Image data affects to next mission phases, they can be used as a data set for ML service development, or as a data set used for creating estimates for pest populations, for example. The robot log data can be used for robot maintenance or for calculating the carbon footprint of the crop. All the possible uses of data are impossible to know but these examples show already that they have value and are worth of collecting and linking to the mission itself.

The MCC uses the mission file to link created data to the mission, and at the same time the mission file transforms to the mission report. The plan of each mission phase contains a placeholder for a link to mission results and these placeholders are filled with actual links to report files at the end of each phase.

As seen in Figure 45, a mission phase has three options: fleet task, service task or user task. Both service and user task are straight forward executions of external activities that require only the sharing of input data and receiving the results data from MCC. The fleet task is more complex. It consists of fleet monitoring tool activations, robot system set up, communication configurations, execution of robot tasks, and transferring of result data to be available for mission reporting. During the mission execution the fleet controller controls the fleet and robots together with individual robot control systems, understanding that in the ground

robots of some pilots (see pilot 1) the individual robot control module is integrated in the robot itself, which follows the task plan defined by the task planner.

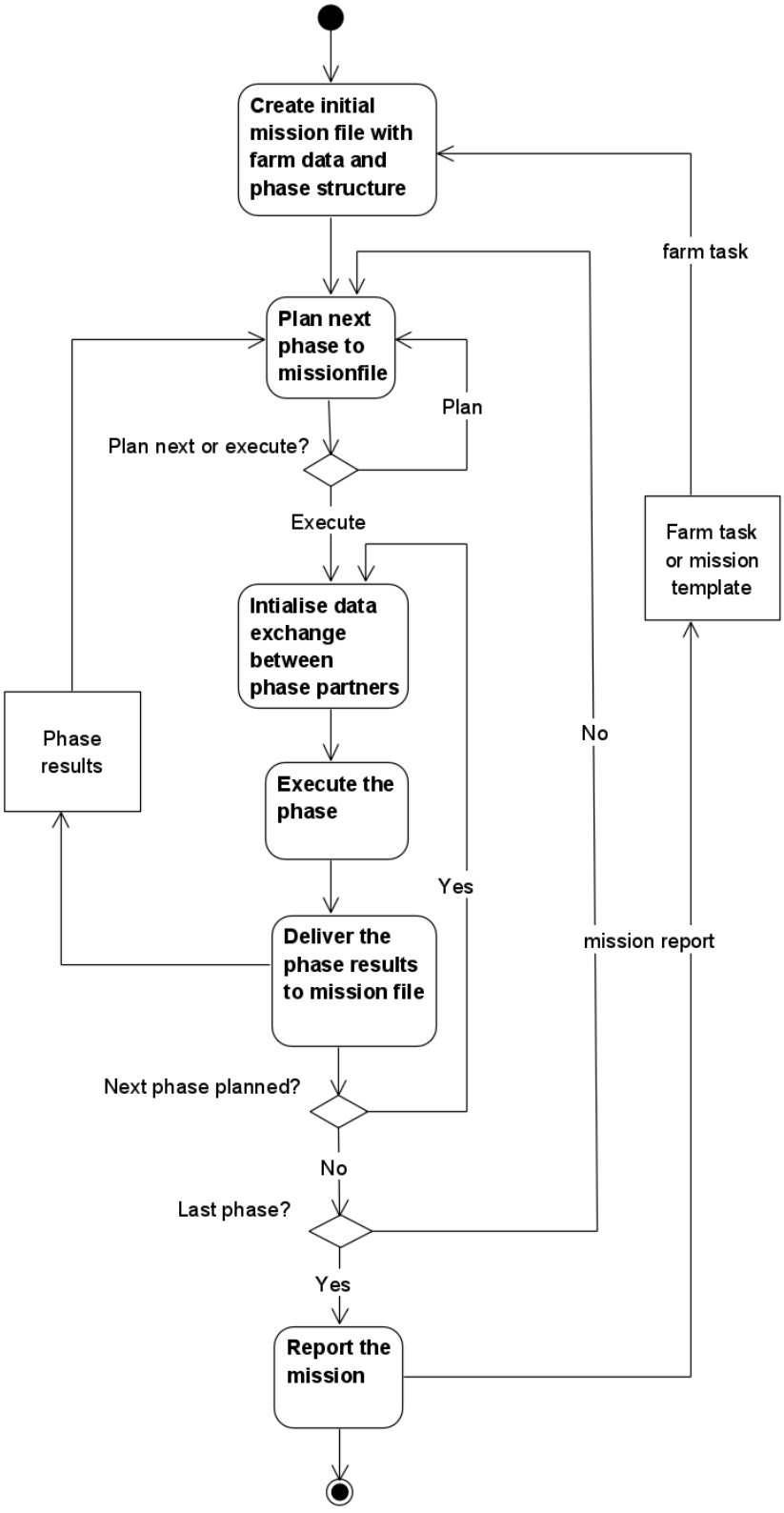

Figure 44 Activity diagram of mission planning and execution

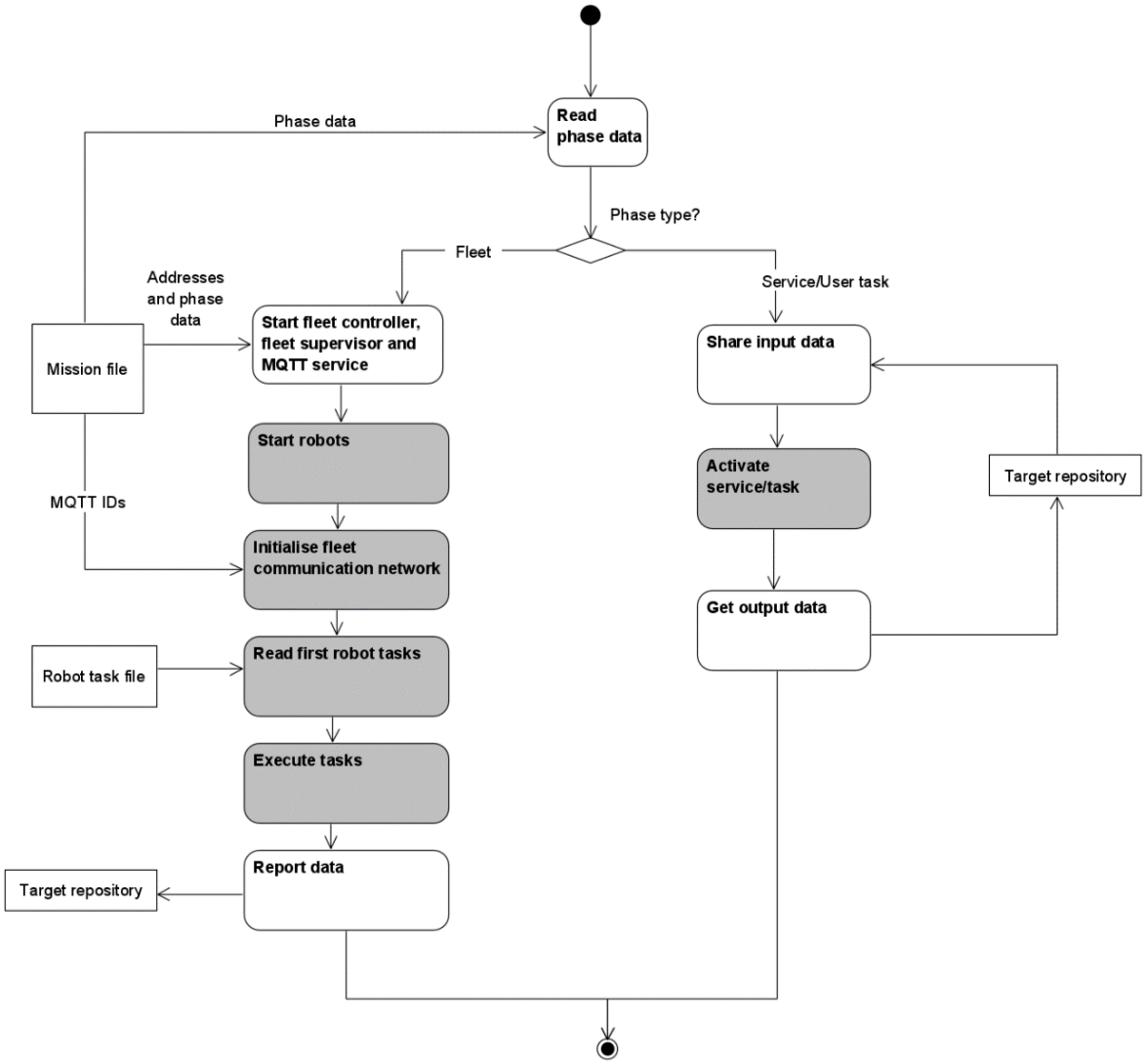

Figure 45 Activity diagram for mission phase execution

6.2.3 Requirements for external systems

The MCC is not an isolated and independent system. It has interfaces to Farm Management Information Systems, various service providers services, users' own tools, and robot operators ICT systems. Therefore, it also sets up requirements for them and to the robots that are used in the missions.

The main requirement for farm management information system and external services is that data between them and MCC can be exchanged through the data space or other data sharing system. The actual implementations depend on system deployments.

Robot control systems must be capable to create MCC status messages and to understand MCC control messages, and to communicate using MQTT. Reference implementation of robot's MCC interface is in FlexiGroBots GitHub.

MCC communication protocol and message formats are not natively supported by any drone as such even if MAVlink-based tools are used as a basis of the mission control centre. DJI Drones do not use MAVlink or other open standard protocols but proprietary one. DJI drones are important to be supported as they are the most widely used in the drone market. DJI provides API that has been used to create a custom DJI drone robot controller. We have implemented such an Android-based controller in the FlexiGroBots project that can take MCC commands and create MCC status messages to perform tasks. The tool is specific to the situational awareness use case but same principles and enables the utilization of low-cost consumer-grade drone use.

6.3 Data models

The main data model related to MCC is the mission file. The mission file is specified as a JSON schema. The initial structure of Mission File is given in Figure 46.

6.3.1 Communication protocols

The MCC communication will be based on IDSA data space protocol and MQTT messaging [4]. The IDSA communication is described in [52]. IDSA protocol is used in the exchange of data files related to mission execution between the companies and organizations. The status and command messages between are described as payloads of MQTT messages.

6.3.1.1 MCC status message

The MCC status messages are the messages send by the robots to the fleet controller. Purpose is to convey the locations and status of the device to the fleet operator and other services that need situational information from the robots. The structure of the messages is following:

MQTT topic is: /<api_key>/<robot_id>/attrs

Where:

- <api_key> is a unique identifier of the mission (MissionID)
- <robot_id> is a unique identifier for the robot
- attrs is topic for robots to publish their status

The MQTT payload conforms to UltraLight protocol where fields are separated with "|" character. The field values are strings. In status message the payload has following fields available:

- “lat”; latitude as GPS coordinates
- “lon”: longitude as longitude GPS coordinates
- “ele”: elevation as meters
- “h”: heading as degrees

- "v": velocity as meters per second
- “stat": status code of sender

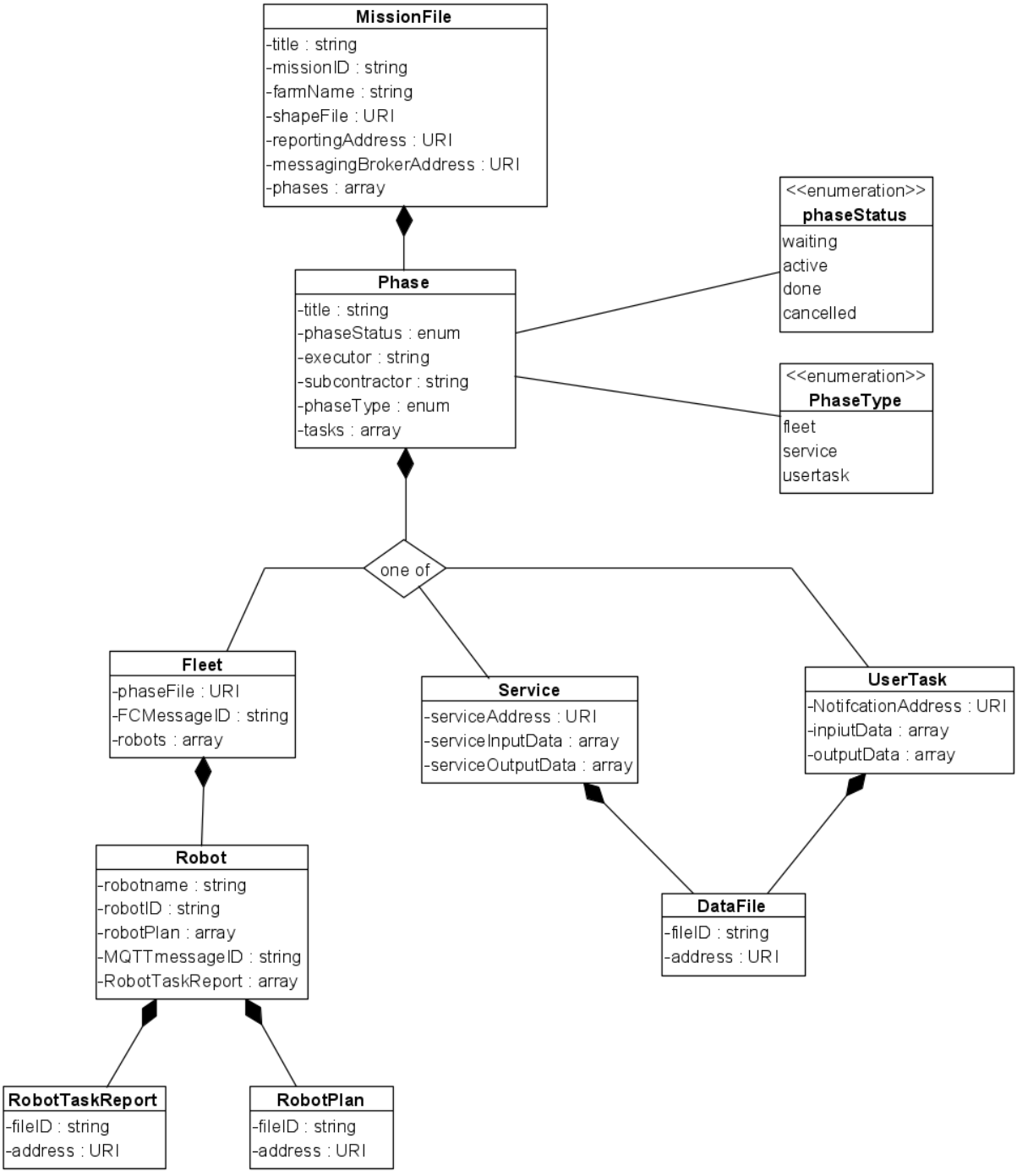

Figure 46 Structure of the mission file

Example payload would be: lat|65.0234|lon|25.2745|ele|5.121|h|43.123|v|5.002|stat|2

Status codes are numbers (presented as strings) from 000 to 999. The status code descriptions and actions are given in a separate table to found at GitHub. Example status codes are:

- 000 = Idle
- 100 = Ready
- 200 = Normal operation
- 300 = Paused
- 400 = Task completed
- 500 = Changing task
- 600 = Task aborted
- 700 = Alert
- 800 = Error

6.3.1.2 MCC Command message

The MCC command message is a MQTT payload meant for Fleet controller to give commands to the robots.

The MQTT topic is: /<api_key>/<field contoller_id>/attrs

Where:

- <api_key> is a unique identifier of the mission (MissionID)
- <field_controller_id> is a unique identifier for the field controller that sends the messages
- attrs is topic for robots to publish their status.

The MQTT message payload conforms to UltraLight protocol where fields (strings) are separated with "|" character. In control message the payload has following fields available:

- "target": receivers “MQTT_id” or “all” for messages to all subscribed devices
- “com”: command code
- “data”: data code

Example payload would be: target|all|com|100|data|000

Example message: /01001/fc_oo1/attrs target|all|comm|100|data|000

Command codes are numbers from 000 to 999. Current codes defined are:

- 100: start task - starts the current active task of the
- 200: stop task - stops the current active task of the robot
- 300: pause task - stops the execution of the current
- 400: resume task - resumes to execute paused tasks
- 500: change task - robot loads a new task identified by task number and makes it to an active task.

Data codes depend on the command. Currently only change task has a data field and it contains the position of the next task in the robot task list to be executed by the robot.

Additional status codes could be defined to commend message table.

6.4 Application Programming Interfaces (APIs)

The MCC is a data driven system and the operation of MCC mission management is based on data and file sharing instead of APIs. The fleet management is based on MQTT messaging.

The data exchange through the data space is done using Data Space Connector's API described in Chapter 3.4. It is used when mission operator and mission manager exchange mission data with an external service or a robot fleet operator. The sequence diagram is shown in Figure 47. The process starts with co-operation agreement between mission and service operator. This is followed by the planning of the phase in which the service is needed. When the plan is ready mission manager must share the mission file data to the service operator, who needs to add the link to the service output to the mission description. Then both parties subscribe to the data artifacts that they will either need for service or receive as a result. During the phase execution the mission operator shares the input data, the data space notifies the service operator (as it has subscribed to data), the service operator executes the service and shares the output data to mission operator, who gets notified by the data space connector.

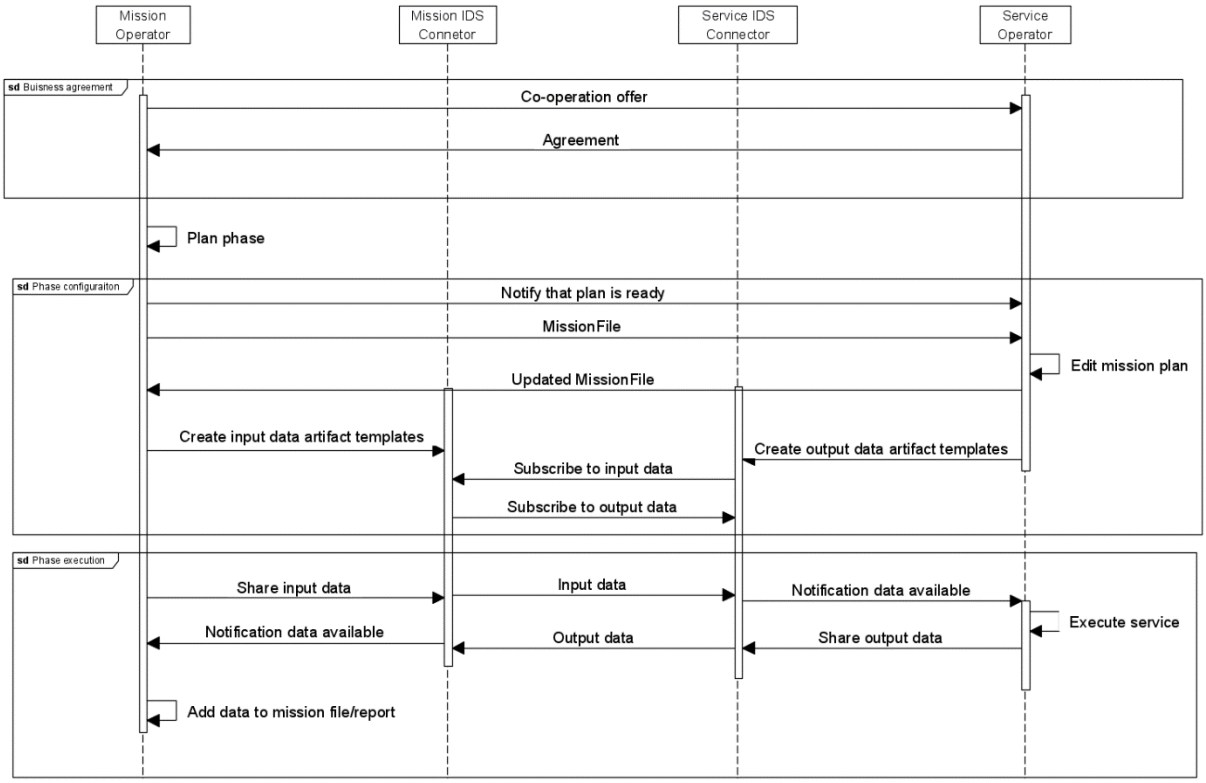

Figure 47 Data exchange between Mission manager and service through data space

The process can be implemented manually using the data space connector interfaces. Automation possibilities and integration of system configurations to mission planner are under study.

6.5 Graphical User Interfaces (GUIs)

The graphical user interfaces for MCC are under development. The Mission manager interface will be a REACT interface based on Mission file schema (Mission Workflow Planner) and the needed functionalities in Mission Workflow Controller and Reporter.

Fleet controller and supervisor will be implemented as an extension to QGroundControl. The user interface will follow the principles of QGroundControl.

A simplified version of Fleet controller and supervisor has been developed using Python 3.9.9 and Tkinter UI library. The focus has been on demonstration of the communication principle and on supporting the development of messaging and file formats. Example of user interface is given in Figure 48. We also developed a simple robot simulator shown in Figure 48 for speeding up the development of messaging design. The simulator was written in Python, and it implements only basic movement of robot through waypoints, creation of status messages, and responses to control messages. In more complete simulation, we plan to use the more complete functional simulators presented in earlier deliverables.

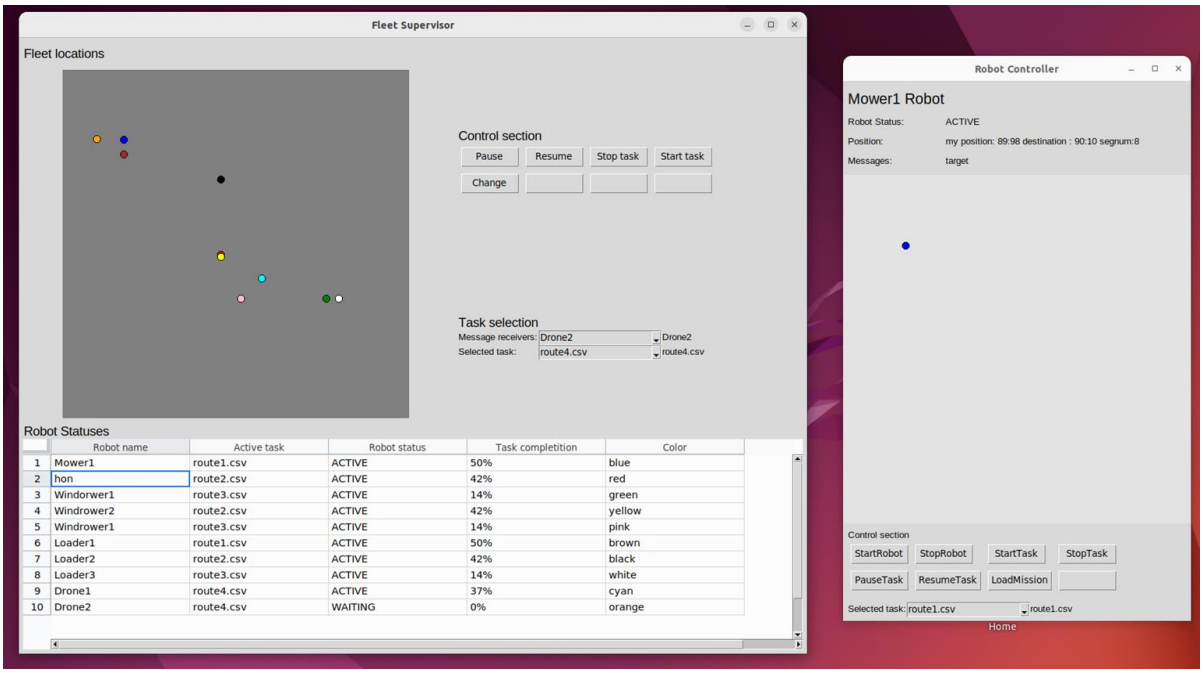

Figure 48 Screenshot from fleet supervisor and controller prototype and a robot simulator

The robot task planner consists of a series of functions written in Matlab and accessed through a main window (Figure 49). This main window is the interface of the planner and has also been developed in Matlab. Through this interface any of the different aspects necessary to calculate a planning can be configured.

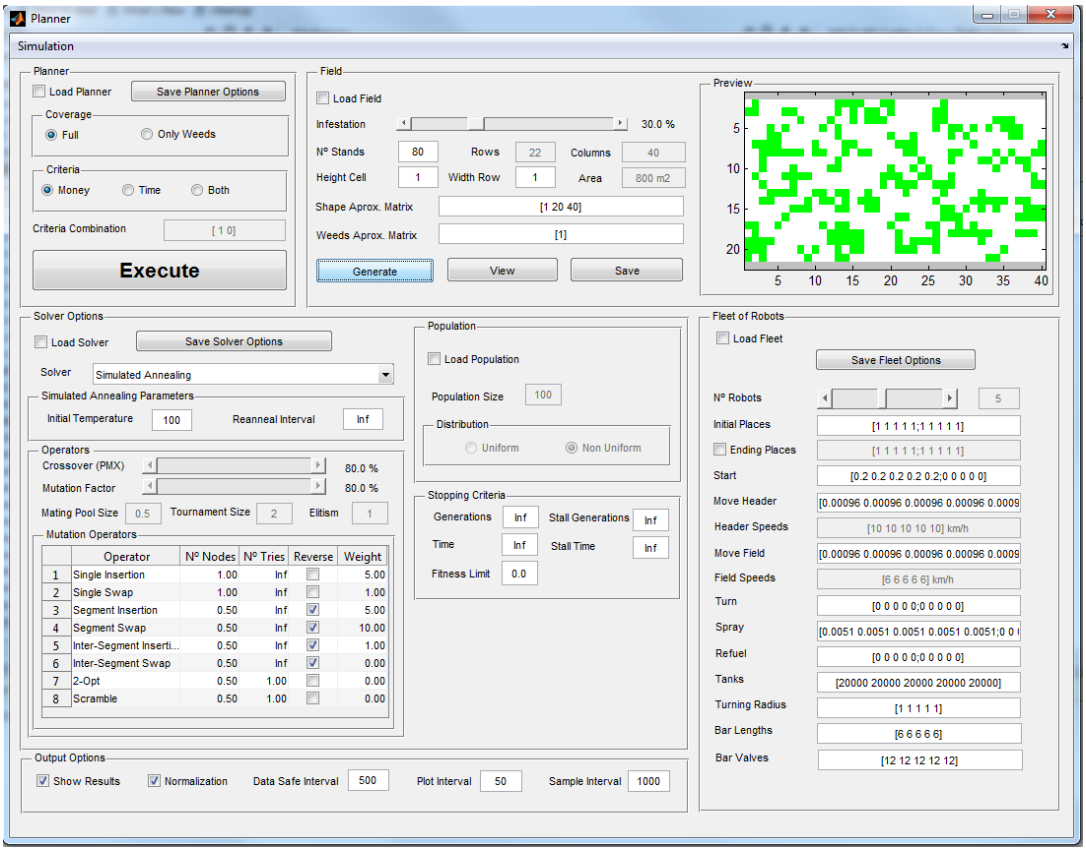

Figure 49 Main window of the robot task planner

A user and developer's guide of the robot task planner is being prepared for delivery with the planner code.

6.6 Installation

Status is that implementations are as a source code in GitHub folders and when available the Docker containers need to be built by user according to the instruction in GitHub.

6.7 Prototype availability within FlexiGroBots

The Mission Control Centre will be available in FlexiGroBots GitHub repository: FlexiGroBots-H2020/Mission-Control-Centre: Mission Control Centre for heterogenous multi-robot operations (github.com)

- The MCC status and command message specifications are in MCC_messaging folder.
- The mission file JSON-Schema in in MissionManager folder.
- The prototype fleet controller and supervisor are in PoC folder.

QGroundControl station has been cloned from its original GitHub to: FlexiGroBots-H2020/qgroundcontrol: Cross-platform ground control station for drones (Android, iOS, Mac OS, Linux, Windows) (github.com)

MQTT messaging with QGroundControl is available in FlexiGroBots-H2020/MVSE: Multi PX4 Vehicle Simulation Environment with MAVLink Communication over MQTT (github.com)

DJI drone controller was implemented for Android OS that is distributed with Apache 2.0 license. Source code of the controller (FGBTrackerI) is hosted at GitHub repository: https://github.com/karikolehmainen/FGBTracker

Tool has been created specifically for Mavic Air 2 which is entry level consumer video drone. Other DJI drones can be easily added to the tool but better DJI drones support natively the type of actions that the tool is created for (following GPS coordinates) so the control sequence created for Air 2 is overly complicated for better and more expensive drones.

Robot task planning tools will be made available to project's GitHub.

6.8 Release planning

The Mission control centre development is at implementation phase and is best followed according to the implementation architecture model. The planned release schedule is given in Table 19 and the user stories of MCC are given in Table 20. The implementation of components is straightforward. The complexity comes from deployment alternatives and the distributed nature of the overall system.

Component	Work to be done	Complexity (1-10)	Estimated release
Mission workflow planner	Finalisation of mission editor, interfacing to robot data and services available for mission, and phase data management in MCC	4	M27
Mission workflow controller	Implementation of phase status changes, phase approval process, and phase activation	4	M28
Mission reporter	Implementation of final report notification	2	M28
Mission repository	Setting up the MCC run-time environment	1	M26
Mission management GUI	Development of REACT UI for MCC services	3	M27
Data space interface	Development of a back-end that provides an interface to data space connector and provides authentication, notification, and web server for applications	3	M28
Fleet controller	Transferring to prototype functionality to QGroundControl extension	3	M27
Fleet supervisor	Adding protocol changes to QGC MQTT interface	1	M26

Component	Work to be done	Complexity (1-10)	Estimated release
MQTT broker	Setting up run-time environment	1	M26
MCC communication	Testing of data space interfaces with MCC components, testing of notifications	1	M25
Robot task planner	Check performance for all pilots and adjust suit all use cases	5	M28

Table 19 Release plan of MCC components

The complete MCC will be ready in M28. The pilot demonstrations are expected to start at M29-M30 giving some time for system set-ups and testing.

User story ID	User story	Priority	Story points
MCC_US_01	Create a simulation environment for development purposes	High	1
MCC_US_02	Create controllers with ground robots in pilot 1 to send missions	High	5
MCC_US_03	Create controllers with ground robots in pilot 2 to send missions	High	5
MCC_US_04	Create controllers with ground robots in pilot 3 to send missions	High	5
MCC_US_05	Create supervisors with ground robots in pilot 1	High	2
MCC_US_06	Create supervisors with ground robots in pilot 2	High	2
MCC_US_07	Create supervisors with ground robots in pilot 3	High	2
MCC_US_08	Implement planner to support missions with multiple robots	Medium	13
MCC_US_09	Implement events processing and alarms notification	Medium	8
MCC_US_10	Implement Data Space connector	Low	3
MCC_US_11	Implement MQTT communication extension for QGC	High	8
MCC_US_12	Create a model for complete mission workflow and its phases	High	8
MCC_US_13	Support multi-partner execution of mission	High	12
MCC_US_14	It should be possible to share data created in robot missions with AI development platform	High	12

Table 20 User stories for MCC

7 Conclusions

This document compiles the progress regarding the FlexiGroBots platform during the second year of the project. It also includes the work planned for the last year of the project, detailing the pending aspects of those tasks that theoretically have already come to an end.

The final version of the Artificial Intelligence platform prototype –which relies on Kubeflow and MinIO– has been developed comprising the storage data lake, ML pipelines tools and hyperparameter tunning functionalities. Moreover, the platform supports GPU workloads and enables the use of APIs to interact with the different components. Pending implementations, such as the integration with the data space and pilots, will be released in the coming months.

IDSA open-source building blocks implemented in the D3.1 has been modified to run in a production-ready solution, Kubernetes. For this purpose, all manifests of each DS component have been developed, and finally, the system has been deployed in a real production cluster. This cluster is located in the cloud providing access to different DS components such as Broker, Omejdn, and connectors. Since the last deliverable, IDSA has updated several components (connectors, broker, Omejdn). For this reason, those components with older versions have been upgraded. Finally, three DS connectors with their certificates have been distributed among the pilots. These connectors are ready to be deployed and automatically connected to the DS.

The outcomes and progress carried out in task "T3.3 Geospatial enablers and services" can be seen as a further step for the establishment of the FlexiGroBots Agricultural Data Space, by facilitating the management and access to the heterogeneous EO data sources in the project pilots, in particular the datasets and derived products from the UAV. In this regard, as the project and its pilots progress, more and more EO datasets will be made available (both from Sentinel 2 satellites and the drones), which would be difficult to efficiently handling and integrating them with other subsystems. Thus, the new implemented features allow to easily generate the STAC metadata necessary for the indexation of these EO datasets into the ODC and later accessed and visualization using the OGC interfaces offered by the data cube).

Different common applications built on top of state-of-the art components have been developed and are presented in this document in three groups based on their goal and scope: situational awareness (SLAM, People detection, location, and tracking; People action recognition, and Moving objects detection); utilities to meet pilot's specific requirements (Orthomosaic assessment, Anonymization, and Dataset generation); and general-purpose applications for the agricultural sector (Disease detection, Pest detection and Weed detection). All these applications are developed maximizing versatility and applicability in different contexts and, potentially, in other related projects. Not all of them have reached the same level of performance yet. Therefore, the development tasks will continue for the next few months to achieve production ready solutions and for their integration in the pilots.

The MCC v1 contains a protype tool for fleet supervision and control. The tool is based on MQTT messaging and to the defined MCC communication messages for robot statuses and fleet controller commands. The Matlab prototype of robot task planner has been developed. The overall concept of mission management has been defined and specification v1 of mission description exists. The path towards v2 contains the development of mission manager services and REACT user interfaces, the integration of fleet management components to QGroundControl software and its user interface, and development of support for data sharing using FlexiGroBots data space. The estimated release of MCC v2 reference implementation is M28 (end of April, 2023).

References

[1] "FlexiGroBots D3.1 – FlexiGroBots patform".

[2] "AI4EU Marketplace," [Online]. Available: https://aiexp.ai4europe.eu/#/marketPlace#marketplaceTemplate.

[3] "AI on Demand," [Online]. Available: https://www.ai4europe.eu/.

[4] "MQTT Version 5.0. Edited by Andrew Banks, Ed Briggs, Ken Borgendale, and Rahul Gupta. 07 March 2019. OASIS Standard. https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html. Latest version: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html".

[5] European Commision (2019), "Ethics guidelines for trustworthy AI | Shaping Europe's digital future," [Online]. Available: https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai. [Accessed 23 01 2023].

[6] European Commision (2020), "Assessment List for Trustworthy Artificial Intelligence (ALTAI) for self-assessment | Shaping Europe's digital future," [Online]. Available: https://digital-strategy.ec.europa.eu/en/library/assessment-list-trustworthy-artificial-intelligence-altai-self-assessment. [Accessed 23 01 2023].

[7] Mitchell, M. et al., "Model Cards for Model Reporting," in *Proceedings of the Conference on Fairness, Accountability, and Transparency, pp. 220–229. doi:10.1145/3287560.3287596*, 2019.

[8] Gebru, T. et al., "Datasheets for Datasets," arXiv. Available at: http://arxiv.org/abs/1803.09010 (Accessed: 29 May 2022), 2021.

[9] "Kubeflow," [Online]. Available: https://www.kubeflow.org/. [Accessed 9 Nobember 2022].

[10] "Protocol Buffers Documentation," [Online]. Available: https://protobuf.dev/overview/.

[11] "Official MinIO web-page," [Online]. Available: https://github.com/FlexiGroBots-H2020/AI-platform/tree/master/minio.

[12] "MinIO library," [Online]. Available: https://github.com/FlexiGroBots-H2020/AI-platform/blob/25-minio_library/minio/Minio_lib.ipynb.

[13] "MinIO repository in FlexiGroBots," [Online]. Available: https://github.com/FlexiGroBots-H2020/AI-platform/tree/master/minio.

[14] D3.1 FlexiGroBots Platform v1.

[15] "International Data Spaces Association GitHub repository," [Online]. Available: https://github.com/International-Data-Spaces-Association.

[16] https://kubernetes.io/es/.

[17] https://github.com/FlexiGroBots-H2020/Data-Space.

[18] "The International Data Spaces Association on GitHub," [Online]. Available: Available: https://github.com/International-Data-Spaces-Association/idsa.

[19] "Official web page of Swagger," [Online]. Available: https://swagger.io/.

[20] "Redis," [Online]. Available: https://redis.io/. [Accessed 24 01 2023].

[21] Flask, "Flask API-REST engine.," 25 01 2023. [Online]. Available: https://palletsprojects.com/p/flask/. [Accessed 25 01 2023].

[22] "Jinja," [Online]. Available: https://jinja.palletsprojects.com/en/3.1.x/. [Accessed 07 02 2023].

[23] https://itsdangerous.palletsprojects.com/en/2.1.x/, "itsdangerous," [Online]. Available: https://itsdangerous.palletsprojects.com/en/2.1.x/. [Accessed 07 02 2023].

[24] "Traefik official web Page," [Online]. Available: https://traefik.io/.

[25] "Internet Society, TLS Basics," [Online]. Available: https://www.internetsociety.org/deploy360/tls/basics/. [Accessed 09 03 2023].

[26] "Dataspace connector Github project," [Online]. Available: https://github.com/International-Data-Spaces-Association/DataspaceConnector.

[27] "The International Data Spaces Association on GitHub," [Online]. Available: https://github.com/International-Data-Spaces-Association/idsa.

[28] "Kubectl official web page," [Online]. Available: https://kubernetes.io/es/docs/tasks/tools/.

[29] "Preconfiguration Postman collection K8s," [Online]. Available: https://github.com/FlexiGroBots-H2020/Data-Space/blob/main/k8s/TestbedPreconfigurationK8s.postman_collection.json.

[30] S. Shinya, S. Mikiya and S. Ken, "OpenVSLAM: A Versatile Visual SLAM Framework," 10 10 2019. [Online]. Available: https://arxiv.org/abs/1910.01122.

[31] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel and J. D. Tardós, "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM," 23 4 2021. [Online]. Available: https://arxiv.org/abs/2007.11898.

[32] stella-cv, "Github stella_vslam," stella-cv, 31 Jan 2021. [Online]. Available: https://github.com/stella-cv/stella_vslam. [Accessed 29 Nov 2022].

[33] Stereolabs, "Stereolabs Zed 2i stereo camera home page," Stereolabs, [Online]. Available: https://www.stereolabs.com/zed-2i/. [Accessed 29 Nov 2022].

[34] X. Zhou, R. Girdhar, A. Joulin, P. Krähenbühl and I. Misra, "Detecting Twenty-thousand Classes using Image-level Supervision," FAIR, 29 Jul 2022. [Online]. Available: https://arxiv.org/abs/2201.02605. [Accessed 29 Nov 2022].

[35] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger and I. Sutskever, "Learning Transferable Visual Models From Natural Language Supervision," FAIR, 26 Feb 2021. [Online]. Available: https://arxiv.org/abs/2103.00020. [Accessed 29 Nov 2022].

[36] N. Wojke, A. Bewley and D. Paulus, "Simple Online and Realtime Tracking with a Deep Association Metric," 21 Mar 2017. [Online]. Available: https://arxiv.org/abs/1703.07402.

[37] roboflow-ai, "Github Zero-shot Object tracking," roboflow-ai, 23 Aug 2021. [Online]. Available: https://github.com/roboflow-ai/zero-shot-object-tracking. [Accessed 14 Nov 2022].

[38] R. Ranftl, A. Bochkovskiy and V. Koltun, "Vision Transformers for Dense Prediction," 24 Mar 2021. [Online]. Available: https://arxiv.org/abs/2103.13413. [Accessed 27 Oct 2022].

[39] M. Contributors, "OpenMMLab's Next Generation Video Understanding Toolbox and Benchmark," open-mmlab, 2020. [Online]. Available: https://github.com/open-mmlab/mmaction2. [Accessed 21 Nov 2022].

[40] C. Gu, C. Sun, D. A. Ross, C. Vondrick, C. Pantofaru, Y. Li, S. Vijayanarasimhan, G. Toderici, S. Ricco, R. Sukthankar, C. Schmid and J. Malik, "AVA: A Video Dataset of Spatio-temporally Localized Atomic Visual Actions," Google, 30 Apr 2018. [Online]. Available: https://arxiv.org/abs/1705.08421. [Accessed 29 Nov 2022].

[41] P. Zhu, L. Wen, X. Bian, H. Ling and Q. Hu, "Vision Meets Drones: A Challenge," 4 Oct 2021. [Online]. Available: https://arxiv.org/pdf/2001.06303.pdf. [Accessed 22 Nov 2022].

[42] Ultralytics, "YOLOv5," Ultralytics, 28 Oct 2021. [Online]. Available: https://github.com/ultralytics/yolov5. [Accessed 27 Nov 2022].

[43] OpenDoneMap, " A command line toolkit to generate maps, point clouds, 3D models and DEMs from drone, balloon or kite images.," OpenDoneMap, 20 Oct 2022. [Online]. Available: https://github.com/OpenDroneMap/ODM. [Accessed 21 Oct 2022].

[44] S. Vélez, C. Poblete-Echeverría, J. Rubio, R. Vacas and E. Barajas, "Estimation of Leaf Area Index in vineyards by analysing projected shadows using UAV imagery,," *OENO One,* vol. 55, no. doi: 10.20870/oeno-one.2021.55.4.4639, pp. 159-180, Nov. 2021.

[45] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment.," *Linux journal,* vol. 239, p. 2, 2014.

[46] H. Hukkelas, R. Mester and F. Lindseth, "DeepPrivacy: A Generative Adversarial Network for Face Anonymization," in *Advances in Visual Computing*, Springer International Publishing, 2019, pp. 565 - 578.

[47] R. Beaumont, "Easily compute clip embeddings and build a clip retrieval system with them," 1 Dec 2020. [Online]. Available: https://github.com/rom1504/clip-retrieval. [Accessed 4 Nov 2022].

[48] C. Schuhmann, R. Vencu, R. Beaumont, T. Coombes, C. Gordon, A. Katta, R. Kaczmarczyk and J. Jitsev, "LAION-5B: a new era of open large-scale multi-modal datasets," Laion, 11 Nov 2022. [Online]. Available: https://laion.ai/blog/laion-5b/. [Accessed 29 Nov 2022].

[49] R. Beaumont, "Easily turn large sets of image urls to an image dataset. Can download, resize and package 100M urls in 20h on one machine.," 29 Jul 2021. [Online]. Available: https://github.com/rom1504/img2dataset. [Accessed 26 Nov 2022].

[50] B. E. Moore and J. J. Corso, "FiftyOne: The open-source tool for building high-quality datasets and computer vision models," Voxel51, 2020. [Online]. Available: https://github.com/voxel51/fiftyone. [Accessed 29 Nov 2022].

[51] S. Haug and J. Ostermann, "A Crop/Weed Field Image Dataset for the Evaluation of Computer Vision Based Precision Agriculture Tasks," in *Computer Vision - ECCV 2014 Workshops*, Springer, 2015, pp. 105-116.

[52] IDSA, "Anforderungen und Referenzarchitektur eines Security Gateways zum Austausch von Industriedaten und Diensten, DIN SPEC 27070:2020-03," 2020.

[53] Conesa-Muñoz, J., Bengochea-Guevara, J. M., Andujar, D., & Ribeiro, A., "Route planning for agricultural tasks: A general approach for fleets of autonomous vehicles in site-specific herbicide applications," *Computers and Electronics in Agriculture,* vol. 127, pp. 204-220, 2016.

[54] Conesa-Muñoz, J., Pajares, G., & Ribeiro, A., "Mix-opt: A new route operator for optimal coverage path planning for a fleet in an agricultural environment," *Expert Systems with Applications,* vol. 54, pp. 364-378, 2016.

[55] K. Deb, "Multi-objective Optimization," in *Search methodologies*, Boston, MA., Springer, 2014, pp. 403-449.

[56] Blum, C.; Roli, A., "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys (CSUR),* vol. 35, no. 3, pp. 268-308, 2003.

[57] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics,* pp. 497-516., 1957.

[58] Shkel, A. M. and Lumelsky, V. , "Classification of the Dubins set," *Robotics and Autonomous Systems,* pp. 179-202., 2001.

Annex A: Model Cards and Dataset Datasheets

Model Card for Model Reporting: MODTL tool				
Based on Mitchell et al. 2019 **With elements from the draft EU AI Act, Annex IV, 2021**				
Category	**Question ID**	**Sub-Category**	**Question**	**Response**
1. Model details. Basic information about the model	1	Person or organization developing model	What person or organization developed the model? This can be used by all stakeholders to infer details pertaining to model development and potential conflicts of interest	ATOS, with partners in the FlexiGroBots project. https://flexigrobots-h2020.eu/
	2	Model date	When was the model developed? This is useful for all stakeholders to become further informed on what techniques and data sources were likely to be available during model development.	Start of development: July 2022 Publication of current version: 5.9.2022
	3	Model version	Which version of the model is it, and how does it differ from previous versions? This is useful for all stakeholders to track whether the model is the latest version, associate known bugs to the correct model versions, and aid in model comparisons.	1.0
	4	Model type	What type of model is it? This includes basic model architecture details, such as whether it is a Naive Bayes classifier, a Convolutional Neural Network, etc. This is likely to be particularly relevant for software and model developers, as well as individuals knowledgeable about machine learning, to highlight what kinds of assumptions are encoded in the system.	The model is a modification of YOLOv5 L, called TPH-YOLOv5. This model improved YOLOv5-L (CNN) based on Transformer Prediction Head for object detection on UAV footage (tiny objects). The model was fine-

Model Card for Model Reporting: MODTL tool				
				tuned on images of tractors and other vehicles from several sources.
	5	Paper or other resource for more information	Where can resources for more information be found?	For more information about the TPH-YOLOv5 base model, see https://github.com/cv516Buaa/tph-yolov5
	6	Citation details	How should the model be cited?	It is not yet available, for the moment only for consortium partners. It will soon be available to the European community.
	7	License and IP	Under which licence is the model published? If necessary, add any other information related to intellectual property (IP).	GPL-3.0
	8	Feedback on the model	E.g., what is an email address that people may write to for further information?	Mario Triviño, mario.trivino@atos.net
2. Intended Use. Use cases that were envisioned during development.	9	Primary intended uses and purpose	This section details whether the model was developed with general or specific tasks in mind (e.g., plant recognition worldwide or in the Pacific Northwest). The use cases may be as broadly or narrowly defined as the developers intend. For example, if the model was built simply to label images, then this task should be indicated as the primary intended use case.	Detection of 5 classes from an UAV point of view: Tractor (class 0); People (class 1); Car (class 2); Van (class 3); Truck (class 4). The primary intended use-case is to detect possible vehicles and people that can be present in the field, in order to avoid dangerous situations.

Model Card for Model Reporting: MODTL tool				
	10	Primary intended users	For example, was the model developed for hobbyists, or enterprise solutions? This helps users gain insight into how robust the model may be to different kinds of inputs.	Intended users are technical providers of agricultural robotics solutions to monitor the position of fleets of vehicles and operators by aerial shot. Farmers can also use the model, but only when integrated into a broader system.
	11	Out-of-scope use cases	Here, the model card should highlight technology that the model might easily be confused with, or related contexts that users could try to apply the model to. This section may provide an opportunity to recommend a related or similar model that was designed to better meet that particular need, where possible. This section is inspired by warning labels on food and toys, and similar disclaimers presented in electronic datasheets. Examples include “not for use on text examples shorter than 100 tokens” or “for use on black-and-white images only; please consider our research group’s full-colour-image classifier for colour images.” Examples include “not for use on text examples shorter than 100	The model is not suitable for imaging heights above 50-80 metres, and its performance suffers greatly as the UAV's flight altitude increases. The ability to detect people is reduced at heights closer to 50 meters.
3. Usage Information	12	Software requirements	What are software requirements and dependencies? If possible, please add a link to an open source repository like GitHub with details on dependencies, the environment and documentation.	The model was trained with the following Python packages on a Linux machine: Python = 3.8.10 PyTorch==1.10 Details are available on GitHub https://github.com/cv516Buaa/tph-yolov5

Model Card for Model Reporting: MODTL tool				
	13	Hardware requirements - Training	What are hardware requirements for training the model (e.g. CPU or GPU)?	The model was trained with an Nvidia T4 GPU with 16 GB RAM. The model was trained in the AI-platform in the FlexiGroBots project. https://flexigrobots-h2020.eu/
	14	Hardware requirements - Inference	What are hardware requirements for deploying the model (e.g. CPU or GPU)? What do users need to take into account regarding hardware regarding deployment and inference?	Inference requires a GPU, it has been tested on an Nvidia GeForce GTX 1070 GPU with 8 GB RAM and gives 6 frames processed per second. If there are no time constraints or requirements a standard CPU is sufficient.
	14	Technical usage Instructions	Provide any other information which helps users use the model. Ideally, add a code snippet illustrating a typical use-case. You can also add a link to a GitHub repository with usage instructions. This is inspired by model cards such as this: https://huggingface.co/microsoft/beit-base-patch16-224-pt22k-ft22k	The model deploy information and code can be found in the original model architecture GitHub repository: https://github.com/cv516Buaa/tph-yolov5

Model Card for Model Reporting: MODTL tool				
	15	Inputs and outputs	Provide a short description of the model's inputs and outputs	The model takes an image as input (1536*1536 pixels) and outputs an image with the detections drawn overlapping the original image
4. Factors. Factors could include demographic or phenotypic groups, environmental conditions, technical attributes, or others.	16	Relevant factors	What are foreseeable salient factors for which model performance may vary, and how were these determined? Model cards ideally provide a summary of model performance across a variety of relevant factors including groups, instrumentation, and environments. For example: What specific instrumentation hardware or software was used to obtain the images, which could influence performance? For which specific environmental conditions was the model designed (e.g. summer in Italy)? Was training and evaluation conducted on specific demographic groups (e.g. mostly images from 18 - 30 year olds in the US)? For more details, see section 4.3 in https://arxiv.org/pdf/1810.03993.pdf	**Images taken from FlexiGroBots Pilot 1** Environmental conditions: The model was trained on aerial images from rapeseed crop fields in Finland during the summer and in day light. Instrumentation: The images were captured with a DJI drone with 4K resolution. Groups: The model is not trained on images of recognizable people. **Other sources (Laion5B & Visdrone)** Environmental conditions: The conditions present in this part of dataset have a high variability and richness of scenarios. Instrumentation: several different unknown cameras. Groups: The model is not trained on images of recognizable people.

Model Card for Model Reporting: MODTL tool				
5. Metrics. Metrics should be chosen to reflect potential real-world impacts of the model.	17	Model performance measures	What measures of model performance are being reported, and why were they selected over other measures of model performance? Please provide all relevant metrics.	F1-score: 0.69 Precision-score: 0.75 Recall-score: 0.64 mAP-0.5: 0.71 Tractor mAP-0.5: 0.64 People mAP-0.5: 0.76 Car mAP-0.5: 0.91 Van mAP-0.5: 0.62 Truck mAP-0.5: 0.57
	18	Decision thresholds	If decision thresholds are used, what are they, and why were those decision thresholds chosen? When the model card is presented in a digital format, a threshold slider should ideally be available to view performance parameters across various decision thresholds.	We recommend a low confidence threshold, such as 0.2, followed by a tracker.
	19	Approaches to uncertainty and variability	How are the measurements and estimations of these metrics calculated? For example, this may include standard deviation, variance, confidence intervals, or KL divergence. Details of how these values are approximated should also be included (e.g., average of 5 runs, 10-fold cross-validation).	The metrics were determined using an 80% - 5% - 15% split in train-val-test split. Hyperparameters were determined on the training set. Adam optimizer. Yolov5 compose lost function.

Model Card for Model Reporting: MODTL tool				
6. Training and Evaluation Data. Details on the dataset(s) used for the quantitative analyses in the card.	20	Datasets	What dataset(s) were used to (1) train and (2) evaluate the model? If possible, please add a link to details on the respective datasets used, for example a datasheet.	The model was fine-tuned on the FlexiGroBots Visdrone_Tractor_v1 dataset. This dataset is composed of images coming from 3 sources: Pilot 1 (Finnish rapeseed field); Visdrone (https://github.com/VisDrone/VisDrone-Dataset); Laion5b (https://laion.ai/blog/laion-5b/)
	21	Motivation	Why were these datasets chosen?	The Visdrone_Tractor_v1 dataset is specifically tailored to the intended use-case of vehicle and people monitoring in agricultural fields.
	22	Data Pre-processing	How was the data pre-processed for evaluation (e.g., tokenization of sentences, cropping of images, any filtering such as dropping images without faces)? Please provide a short description. You can also add a GitHub link to the respective pre-processing scripts.	Images are processed in colour and 1536x1536 pixels. For details, see the pre-processing script here: https://github.com/cv516Buaa/tph-yolov5

Model Card for Model Reporting: MODTL tool				
7. Quantitative Analyses	23	Disaggregated results and fairness	How did the model perform with respect to each factor (see question 15)? Quantitative analyses should be disaggregated, that is, broken down by the chosen factors. Quantitative analyses should provide the results of evaluating the model according to the chosen metrics, providing confidence interval values when possible. Parity on the different metrics across disaggregated population subgroups corresponds to how fairness is often defined. For an example, see figure 2. in https://arxiv.org/pdf/1810.03993.pdf	The model was only evaluated on one environment (rapeseed crop field in Finland) and specific instrumentation. Disaggregated results for different environments and instrumentation are therefore not available.
8. Ethical Considerations	24	Ethical Considerations	Example topics for ethical consideration: Does the training data contain sensitive information? What risks and harms could arise during the use of the model? Which mitigation measures are recommended? Are there particularly problematic use-cases? Did the model go through an ethical assessment procedure?	The training data only contains aerial images of vehicles and people, and no recognizable elements or sensitive information are present. The model itself does not pose risks, as it only detects vehicles. However, the low accuracy can lead to problems when applied to try to avoid collisions or unnecessary approaches between agricultural field elements. Ethical impacts of the model were discussed in the FlexiGroBots project.

Model Card for Model Reporting: MODTL tool				
9. Caveats and Recommendations	25	Caveats and Recommendations	This section should list additional concerns that were not covered in the previous sections. For example, did the results suggest any further testing? Were there any relevant groups that were not represented in the evaluation dataset? Are there additional recommendations for model use? What are the ideal characteristics of an evaluation dataset for this model?	Performance of the model was only tested for the environmental and instrumental factors explained above. If the model is applied in a different context, additional assessments should be conducted. Moreover, we recommend that users assess the model not only on our test data, but also in their own specific use-cases in a specific environment and instrumentation when embedded into a broader system for.

Table 21 Model card: MODTL TPH-YOLO detector

Datasheets for Datasets:			
Based on Gebru et al. 2021 With elements from the draft EU AI Act, Annex IV, 2021			
Category	**Question ID**	**Question**	**Response**
1. Motivation	1	For what purpose was the dataset created? Was there a specific task in mind?	The dataset is designed to be a training dataset for a machine learning detector to obtain the bounding boxes of vehicles and people on agricultural fields from UAV footage.
	2	Who created the dataset (e.g., which team, research group) and on behalf of which entity (e.g., company, institution, organization)?	The dataset was created by ATOS, with the collaboration of FlexiGroBots Finland Pilot 1 partners.

Datasheets for Datasets:			
	3	Who funded the creation of the dataset?	The dataset was created as part of the FlexiGroBots project. The project received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017111.
	4	Any other comments?	Additional information on the FlexiGroBots project is available here: https://flexigrobots-h2020.eu/
2. Composition	5	What do the instances that comprise the dataset represent (e.g., photos of plants, paragraphs from news articles)?	Photos of tractors, people, cars, trucks and vans from an aerial perspective.
	6	How many instances are there in total? (e.g. how many photos)	10174 labelled images.
	7	Is the dataset a sample, or does it contain all possible instances? If it is a sample, then what is the larger set? Is the sample representative of the larger set? Please elaborate.	The dataset comprises photos from three different sources: (1) images obtained in the Finnish rapeseed fields of FlexiGroBots Pilot 1; (2) aerial images of tractors extracted from the freely accessible Laion5b dataset; (3) images from the Visdrone dataset.
	8	Is there a label associated with each instance? If so, please describe the labels. If the data was annotated manually, please describe the coding instructions for annotators.	Images are annotated with detections of five classes: Tractor, People, Car, Truck and Van. The annotations were automatically performed with Detic model on previously unlabelled photos from Laion5b or Pilot 1. Visdrone images came with labels. More information about Detic can be found here: https://github.com/facebookresearch/Detic The annotation format is YOLO format: (class_id, x_centre, y_centre, width, height)
	9	Is there a codebook or more detailed documentation of each variable and meta data in the dataset? If so, please provide a link.	The annotation instructions and explanation of each variable is available in the section "Convert the Annotations into the YOLOv5

Datasheets for Datasets:			
			Format" include in this blog: https://blog.paperspace.com/train-yolov5-custom-data/
	10	Are you aware of any potential errors, sources of noise, or redundancies in the dataset?	The data were taken at different sources in different conditions to increase variability. The automatic labelling process depends on the quality of the annotation results provided by Detic, even though a subsequent manual inspection has been carried out to avoid duplicity or inaccurately labelled objects, the dataset is sure to continue to contain erroneous images.
	11	Does the dataset contain data that might be considered confidential or offensive (e.g., data that is protected by legal privilege or by doctor-patient confidentiality, or offensive images or texts)?	No.
	12	Does the dataset relate to people? E.g. a dataset relates to people if farm workers could be visible on some images.	As the main subject in the images are people and vehicles, people might be visible in the images but not recognizable.
	13	If the dataset relates to people, please elaborate on data protection measures that have been taken. For example, did individuals provide consent? Were they informed about their rights based on the GDPR?	All workers who could be visible provided explicit consent to the collection, processing and publication of the data. The consent followed the standards of the GDPR. As an additional step before publication, images were anonymised with an in-house face blurring software.
	14	Is it possible to identify individuals either directly (e.g. through their faces) or indirectly (i.e., in combination with other data) from the dataset?	No.

Datasheets for Datasets:			
	15	Does the dataset contain data that might be considered sensitive in any way (e.g., data that reveals racial or ethnic origins, or locations or biometric data)?	No.
	16	Any additional information?	The quality of the dataset have been proved enough to train a detector with F1-score: 0.69 at least.
3. Collection Process	17	How was the data acquired? (e.g., hardware apparatuses or sensors, manual human curation, software programs, software APIs)	Images were captured with several devices, some of them unknown. The part coming from the Finnish FlexiGroBots Pilot has been taken with a DJI drone with 4k resolution.
	18	Who was involved in the data collection process (e.g., students, crowdworkers, contractors) and how were they compensated (e.g., how much were crowdworkers paid)?	The data collected in the FlexiGroBots' context was collected by Pilot 1 partners and annotated automatically by Atos. A manual review of the resulting dataset was performed also by Atos. All this done by full-time employees at Atos as part of their regular paid work.
	19	Over what timeframe was the data collected?	July/August 2021
	20	Was any ethical review process conducted?	The creation and processing of the dataset followed the ethical procedures of the FlexiGroBots project. For more details, see Task 1.5 and Deliverable 2.6 on the FlexiGroBots website: https://flexigrobots-h2020.eu/library/deliverables
	21	Any additional information?	NA
4. Pre-processing/ cleaning/ labelling	22	Was any pre-processing/cleaning of the data done (e.g., removal of instances, processing of missing values)?	Yes, the data was cleaned manually and algorithmically to exclude noisy, irrelevant data and anonymise images of individuals.
	23	Any additional information?	NA

Datasheets for Datasets:			
5. Uses	24	Has the dataset been used for any tasks already? If so, please provide a description.	The dataset is currently being used in a pilot of the FlexiGroBots project. The dataset has been used to train a deep learning detection model to detect people and vehicles in agricultural fields and avoid collisions or dangerous approaches. More information will be available at: https://flexigrobots-h2020.eu/
	25	What (other) tasks could the dataset be used for?	The dataset can be used for any use-case which involves the detection of the 5 detection classes from an aerial point of view.
	26	Is there anything about the composition of the dataset or the way it was collected and pre-processed/cleaned/labelled that might impact future uses?	Automatic labelling of a part of the data may result in lower quality of the dataset.
	27	Are there tasks for which the dataset should not be used? If so, please provide a description.	We only recommend using the dataset for the intended use-case described above.
	28	Any other comments?	NA
6. Distribution	29	Will the dataset be distributed to third parties outside of the entity (e.g., company, institution, organization) on behalf of which the dataset was created? If so, how and when?	The dataset will be uploaded in 2023 on the Zenodo platform
	30	Does the dataset have a digital object identifier (DOI)?	NA

Datasheets for Datasets:			
	31	Will the dataset be distributed under a copyright or other intellectual property (IP) license, and/or under applicable terms of use (ToU)? If so, please describe.	The dataset is distributed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 license. Due to the inclusion of Visdrone dataset.
	32	Do any export controls or other regulatory restrictions apply to the dataset or to individual instances? If so, please describe.	No.
	33	Any other comments?	NA
7. Maintenance	34	Who is supporting/hosting/maintaining the dataset?	Mario Triviño, ATOS
	35	How can the owner/curator/manager of the dataset be contacted (e.g., email address)?	mario.trivino@atos.net
	36	If the dataset relates to people, are there applicable limits on the retention of the data associated with the instances (e.g., were individuals in question told that their data would be retained for a fixed period of time and then deleted)?	While the dataset was anonymised, the precautionary consent form filled in by individuals requires a retention limit of 5 years.
	37	Any other comments?	NA

Table 22 Dataset datasheet: Visdrone-Tractor-v1

Annex B: Robot Task Planner

As mentioned above, we start from a task planner developed in previous projects [53] [54]. Work is currently underway to have a communication based on the IDSA data space to obtain the maps generated by the UAVs and to share the generated plans to other parts of the MCC architecture. In addition, modifications are being made to properly adapt the behaviour of the planner to the new scenarios. In the following, some relevant aspects of task planner are summarised.

The problem of finding the plans - which include the routes - to be followed by the ground robots in order to carry out an agricultural task jointly and optimally, is approached from a very general point of view, taking into account a multitude of aspects that can influence the associated plans and routes. These are the characteristics of the fleet (number of vehicles, working speeds, turning radii, tank capacity, working width, etc.), the crop field (shape, number of rows, direction of cultivation, weed distribution, etc.), the type of task (spraying, inspection, task with full or partial coverage, etc.), and the optimisation criteria (distance travelled, cost per input, time spent, etc.). In this approach, the planning problem is formulated in terms of a combinatorial optimisation.

In agriculture, fields are usually sown in contiguous rows so that, as the crop grows, they are formed by parallel rows of plants. Between each two adjacent rows, there is a small furrow or empty space, usually called a lane, running lengthwise across the field, the width of which is determined by the water, light or space requirements of the crop species. Vehicles driving through crops are obliged to move in the direction of cultivation, i.e., the direction in which the plants are arranged, otherwise the crop rows would be stepped on and the plants would be seriously damaged. In addition, the furrows would cause many vibrations that would affect the correct operation of the machinery, in particularly the implements.

Another important restriction is to avoid driving in reverse, as most agricultural implements are not designed to work in reverse. This is the case for ploughs, harrows, etc. Even in the case of those implements that could be reversed because they are suspended (spray booms, atomisers, etc.), there would be a high probability of damaging the crop and, in addition, the extra movement would accentuate soil compaction in the area. Therefore, this work is based on a real situation in which the manoeuvres are always carried out outside the cultivation area, more specifically in the headlands, which are the areas at the ends of the crop in order to allow the turning and movement of the vehicles, and thus the transit between the different areas of the crop. On the other hand, the tasks must be carried out in such a way as to guarantee total coverage of the crop without overlapping. Taking into account the requirements, the representation of a crop that is used divides the field into separate strips, which are hereinafter referred to as tracks as they constitute the transit routes within the

crop. The tracks are parallel and in the direction of cultivation, with a width equal to the working width of the task to be carried out, which will coincide with the implements used.

Therefore, the aim is to cover all tracks exactly once using one of the vehicles in the fleet.

As it is not possible to turn within the tracks (or crop), the routes will be sequences of tracks interspersed with manoeuvres to transit between them. Figure 50 shows a field with 7 tracks, all oriented by construction according to the direction of cultivation and consisting of 5 tramlines (4 inside plus half on each side), because the working width of the tool used, in the example, covers 5 crop rows.

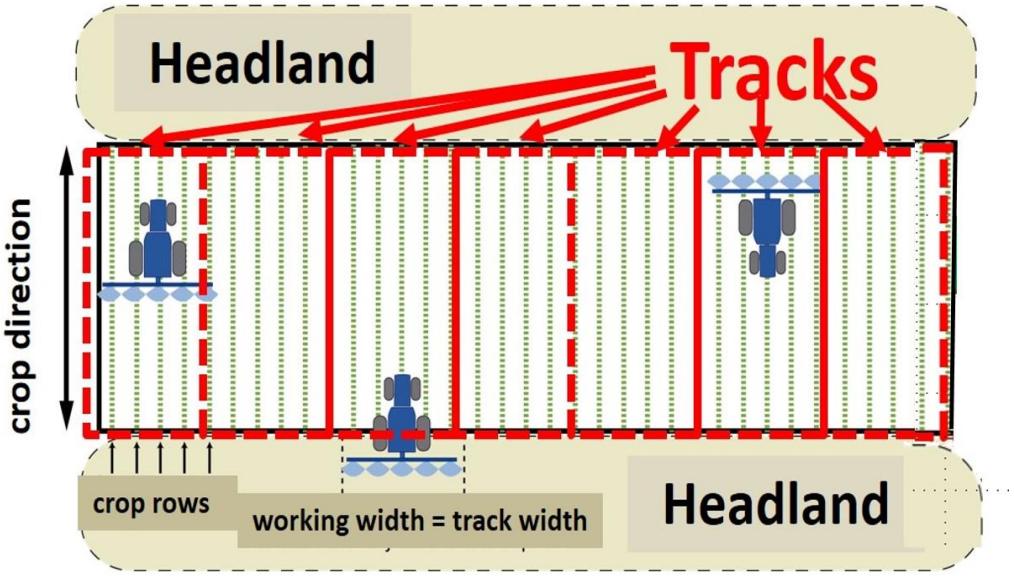

Figure 50 Field divided into 7 tracks of 5 crop rows each

It is important to highlight the generality of the proposed representation, as it allows the representation of irregularly shaped fields of varying direction, such as those shown in Figure 51, using tracks of different lengths and with different shapes to adapt them to the crop direction.

(a) (b)

Figure 51 (a) Irregularly shaped fields with varying crop directions and (b) their representation on tracks

Mathematically, the problem can then be seen as finding the sequence of tracks that minimises the overall cost of the task, since the order in which the tracks are traversed determines the paths to be followed and, in turn, the paths affect the overall cost. This is a combinatorial optimisation problem, in which the optimal sequence must be found within a finite set of possible combinations. Since all tracks have to be travelled exactly once, with only one vehicle, the problem could be formulated as a classical combinatorial optimisation problem, more precisely as a TSP (Traveling Salesman Problem), but without the requirement of the return to the starting track.

However, in this case, the problem has to be approached in a more general way to adapt the approach to the work of a fleet of vehicles, i.e., the problem to be solved would then be similar to the mTSP (multi Traveling Salesman Problem). With this approach, it is possible to obtain those routes that perform total field coverage without overlaps at minimum cost; however, this model still does not consider the carrying capacity of the vehicles to store/transport the inputs. Considering this limitation, in mathematical terms, the problem is similar to the CVRP (Capacitated Vehicle Routing Problem). However, a relevant difference with respect to the CVRP problem makes the real agricultural problem more complex. In the CVRP the vehicles have a certain capacity to store goods that cannot be reloaded in any case, so that one of the preconditions of the problem is that the sum total of the vehicles' capacities is greater than or equal to the sum total of all the customers' demands, otherwise there would be no solution. If this condition were assumed in the agricultural case, it would only be possible to work with fields with a need for inputs less than or equal to that initially transported by the fleet as a whole, i.e., there would be no possibility of resupply. Clearly, this simplification limits the generality of the problem formulated, since regardless of the size of the fleet and the capacity of the vehicles, it is impossible to guarantee that any task can be carried out in any field without resupply. In other words, resupply is an inherent operation of agricultural tasks and as such must be considered in the formulation of the problem.

Extending the classical CVRP problem to consider refuelling considerably increases the difficulty of formulating the problem. A first approximation might assume that it is sufficient to go to the supply depot every time a track runs out and the vehicle does not have enough input to deal with the next track, however, this is not necessarily the best strategy. The optimal time to resupply may vary depending on the location of the supply depot, the amount of input remaining in the vehicle, and the amount required to complete the task. For example, if the depot is close by when finishing a track, it may be worthwhile to resupply even if there is sufficient supply to treat the next tracks if the resupply trip from a point further away from the field can be avoided. On the other hand, it can happen that even with a low supply it is not necessary to go to the depot if the tracks are redistributed among the vehicles of the fleet in such a way that the vehicle concerned has enough with the quantity it is carrying.

As the tracks are treated individually, refuelling can only take place during transitions from one track to the next. Thus, the refuelling sub-problem is reduced to finding the optimal

transitions to and from the depot. Considering refuelling is an extension that significantly increases the complexity of the problem.

Another real situation that can be frequent is that the vehicles do not all have the same characteristics, i.e., the fleet is heterogeneous. Considering this case implies to consider the associated costs of each vehicle. Furthermore, it must also be taken into account the particular capacity associated with each vehicle. When the vehicles are homogeneous, in terms of cost, it is irrelevant for optimisation the specific vehicle with which each track is treated, since all vehicles will perform the same task with the same associated cost. Hence, in this case the optimisation is only in the distance travelled in the transitions to connect the tracks. When considering vehicles with different characteristics this is not the case because it is no longer indifferent, which vehicle is used on which track. For example, a vehicle with a lower fuel consumption will always be preferable to one with a higher fuel consumption. These types of multi-objective problems are solved with methods capable of establishing relationships between solutions with different costs [55].

In summary, the planning problem for carrying out an agricultural task in an optimal way is approached under a broad perspective, obtaining by construction solutions with full coverage and without overlaps that consider: 1) multiple vehicles with different characteristics, 2) the characteristics of the task itself and 3) multiple objectives or optimisation criteria. Also, with the proposed problem approach, the shape of the field does not matter as the performance is decomposed into a set of tracks with the treatment width and following the layout of the crop lines. In short, in all cases the problem is reduced to finding the best order to go through all the tracks regardless of where they are located or the layout they have. It is also important to note that tasks requiring partial coverage (e.g., a pest control treatment where you only want to go to infested tracks) are a sub-problem of those using full coverage (instead of considering all tracks you would consider only the infested ones), and therefore this case is also covered by the proposed formulation.

The most common practice to solve a combinatorial optimisation problem such as the one described, is to use a meta-heuristic algorithm [56].

Three available meta-heuristic algorithms have been evaluated to build the robot task planner, Simulated Annealing, Genetic Algorithms and Non-Dominated Sorting Genetic Algorithm 2 (NSGA-II). These algorithms were selected because they are widely used methods for solving classical route planning problems, such as the aforementioned TSP, mTSP and CVRP.

The three proposed methods share a common strategy: 1) they start from a randomly generated set of solutions X (hereafter also called working set), 2) they evaluate X to determine if any of the termination criteria are fulfilled, e.g., if any solution is good enough or if the maximum search time has been exceeded. If so, the search is terminated and if not 3) the best solutions in X are selected and stored in a new intermediate set Y. 4) new solutions are constructed from those contained in Y and stored in X' (neighborhood set, child solution

set or simply child set). 5) the working set X is replaced by some combination of the sets X and X', and 6) the process is repeated from 2). This operation is shown in Figure 52.

```
1     X = initializeSolutionSet ();
2     WHILE (NOT TerminationConditions (X))
3           Y = selectBestSolutions (X);
4           X'= buildNewSolutions (Y);
5           X = combineOld&NewSolutions (X, X');
6     END
```

Figure 52 Procedure summarising the most external part of the operation of the three selected meta-heuristic algorithms

Representing the solution

Any solution to the agricultural planning problem must contain the information necessary to determine how the tracks are distributed among the vehicles in the fleet, in what order each vehicle must travel the assigned tracks, and whether they must refuel and, if so, among which tracks.

The order of the tracks and their distribution among the vehicles in the fleet can be represented by a permutation of the union of the set P, which contains all the tracks pi into which the field was divided, and the set S, which contains m - 1 separators s_i, where m is the number of vehicles in the fleet. In this way, regardless of where the separators are placed, the permutation will always be divided into m parts, with the first part containing the tracks run by the first vehicle, the second part by the second vehicle, and so on. The order in which the tracks are to be traversed is determined by the order in which they appear in each part, so that you start with the leftmost track and continue with the next track to the right until you reach the last track (the track furthest to the right). If any part contains no tracks (because the permutation contains two separators in a row), it means that the corresponding vehicle is not involved in the planning.

For refuelling, as indicated in the formulation, a binary vector b with as many elements as there are transitions in the permutation will suffice, with the value 1 indicating that there is refuelling during the transition i, and the value 0 indicating that there is not. However, with this representation, the vector should have a variable size since the number of transitions depends on the number of vehicles used in the solution. size would be n - m_u, where n is the number of tracks and m_u is the number of vehicles used. In order not to use variable representations, which are more difficult to implement, it was decided to use a vector with as many components as tracks, redefining the meaning of each b_i in such a way that a 1 means that refuelling must take place between track p_i and the next one (p_i+1), and 0 means that there is no refuelling. In case there is no next track, because p_i is the last one in the route, b_i must be ignored regardless of its value because it makes no sense to perform a refuelling. It

does not make sense to perform a resupply after the task has been completed. With this representation the problem has $2n + m_u$ - 1 decision variables, being n of them binary.

In what follows, we will refer to the first part of the solution as permutation vector, hint vector or simply permutation, while the second part of the solution will be the binary vector of replenishments or simply replenishment vector.

It is important to note that with this representation the solutions by construction cannot give rise to collisions between the vehicles within the crop, as the vehicles always work on different and disjoint tracks, however these can occur at the headlands as it is a common transit place for all vehicles. For this reason, it is assumed that the fleet has a system to avoid collisions at the headlands.

Decoding: getting the actual trajectories and the action plan

Once the tracks in charge of each vehicle have been identified, the order in which they must be travelled and at which points there are refuelling points, all the necessary information is available to obtain the exact trajectories of each vehicle. Within the field, the trajectories are defined by construction to be the inner centre lines of each track connecting the ends, so that the implement passes - and works - along the entire track.

The calculation of trajectories associated with transitions is more complex. Transitions can be either directly between two tracks or by stopping at the depot for refuelling. In the latter case, they can be represented as an outbound transition to the depot plus a return transition. As the position of the depot and the end points of the tracks are well known, the position of the start and end points are known for all transitions. However, as real vehicles cannot turn freely in any direction because they have a minimum turning radius related to their mechanical limitations, it is not possible to connect both points simply by a straight line but it is necessary to know the start and finish orientations. For the case of points on tracks, it is easy to determine the departure/entry orientations from the direction of cultivation; whether the point is a departure or entry point, and whether it is on the upper or lower headland. Equations (3.14) and (3.15) obtain the outbound θ_{out} and inbound θ_{in} orientations for a point p from the growing direction θ_c, assuming that this has been calculated in the bottom headland - top headland direction (see direction of θ_c in Figure 53).

In the case of the depot, the entrance and exit orientation are assumed to be similar to those of the entrance/exit road giving access/exit to the site. Figure 53 shows the entrance and exit orientations of the tracks of a field and the supply depot.

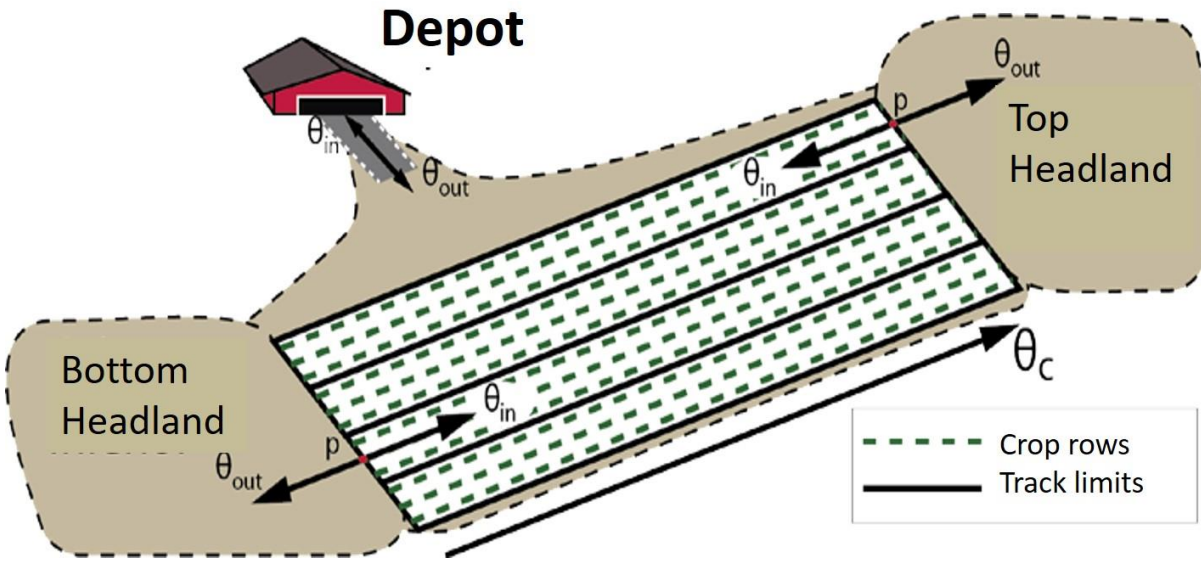

Figure 53 Orientations of a crop

Knowing then all the exit and entry positions and orientations to the crop, the problem of finding transitions with minimum distance translates into finding the shortest path that a vehicle with a turning radius must follow to get from an exit point with an initial orientation to a destination point with a given orientation. According to Dubins' theorem [57], the shortest path between an exit configuration (exit point and orientation) and an entry/arrival configuration (arrival point and orientation) is always a sequence of straight lines (R) and circular arcs (C) of radius $rmin$ of the form CRC, CCC or a subsequence of the latter two, where $rmmin$ is the minimum possible turning radius. Extrapolating to vehicle trajectories, as vehicles can turn either left or right, the two types of possible trajectories are two types of possible trajectories can be broken down into six cases that arise from particularising the circular arcs (C) into left (I) or right (D) turns. The six resulting cases are IRI, IRD, DRI, DRD, IDI, and DID (see Figure 54). In other words, any minimum path between an input and an output configuration will always have a shape that fits one of these six cases or their subcases.

To know which of the cases contains the minimum path given any two configurations, a usual strategy is to evaluate all six cases and keep the one that gives rise to the shortest path. Although [58] showed that by geometric reasoning it is possible to know in advance which case contains the minimum path, thus saving a lot of computational time, which is of great interest if one has to plan in real time or has to evaluate many paths. The latter situation is the one we face in this planner, since meta-heuristic algorithms work by constructing a multitude of solutions that have to be evaluated.

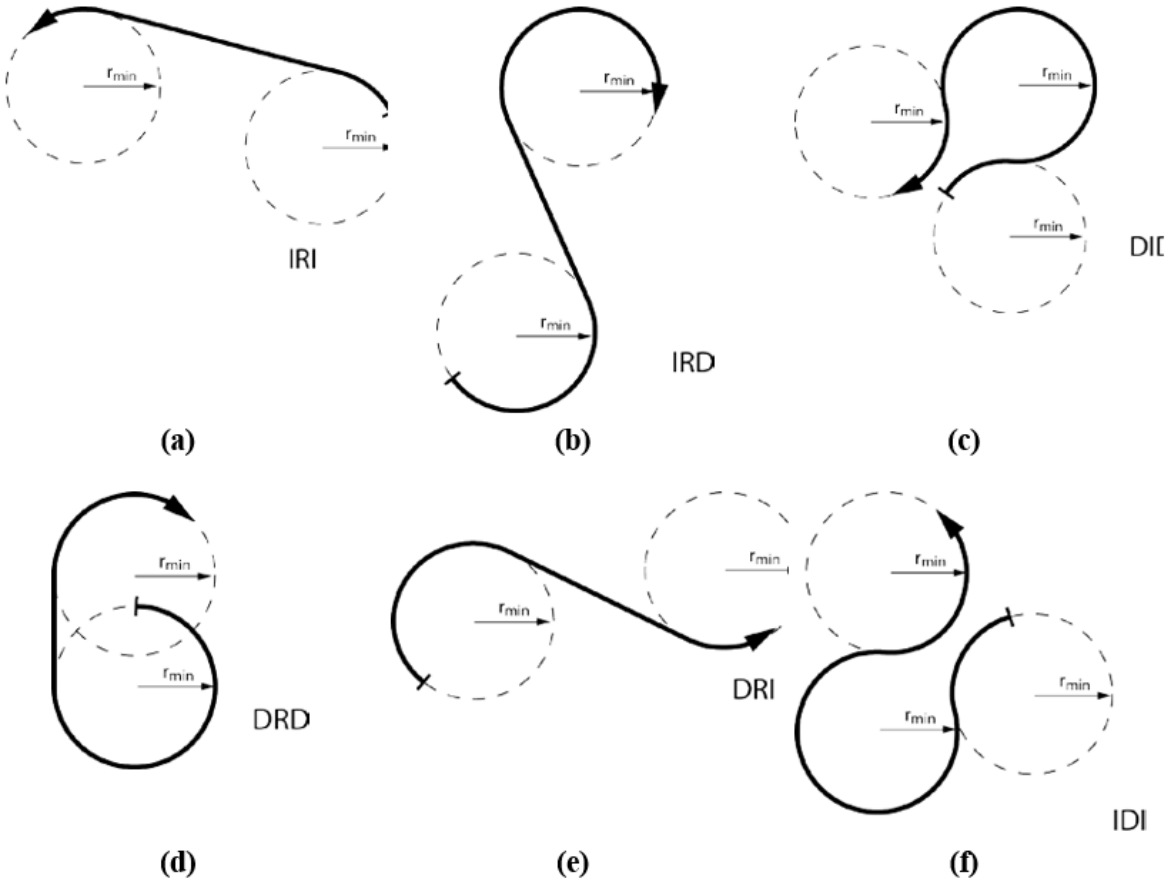

Figure 54 The six possible trajectories according to Dubins' theorem [57] between two points with fixed exit orientation and entry orientation

Therefore, by means of [57] and [58] it is possible to know the shortest path for any transition. However, as shown in Figure 55, these turns can be concretised in ∏ and Ω manoeuvres, when working on rectangular fields and transiting between tracks. For cases where it is necessary to work with more general transitions, such as between tracks and reservoirs or between tracks in non-rectangular fields, the more general procedure described in [58] can be used.

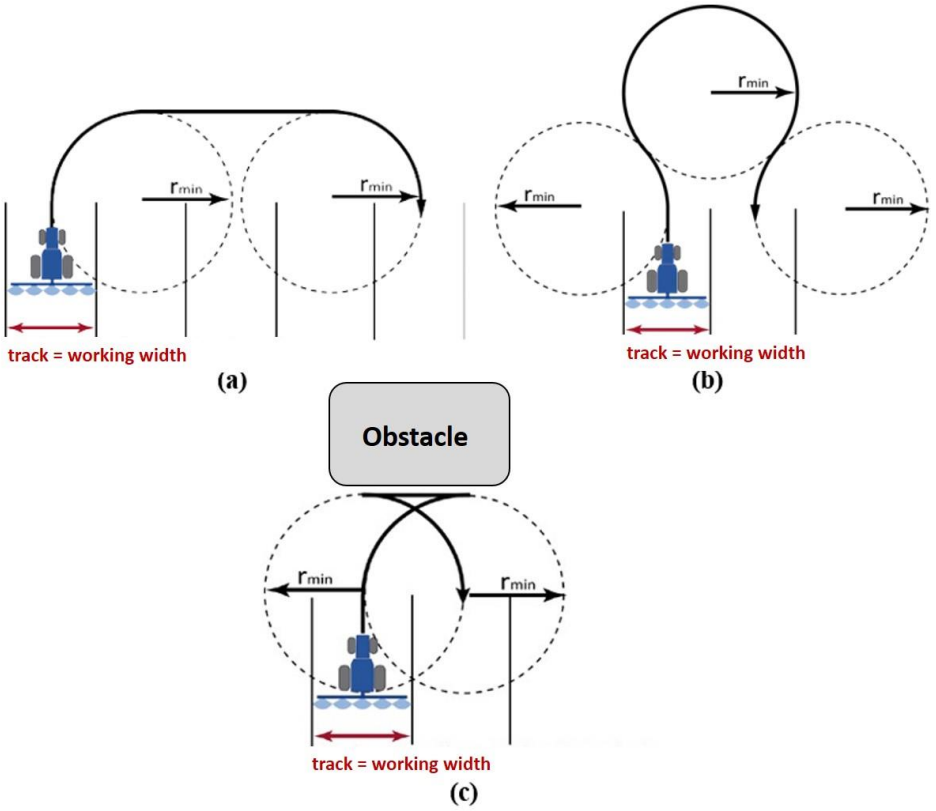

Figure 55 Manoeuvre types for a vehicle of radius r_{min} in a regular field. (a) ∏-turn, (b) Ω-turn and (c) T-turn

$$\prod(i,j) = |i-j|a_c + (\pi - 2)r_{min}$$

$$\Omega(i,j) = r_{min}\left(3\pi - 4\,sin^{-1}\left(\frac{2r_{min} + |i-j|a_c}{4r_{min}}\right)\right)$$

$$turn(i,j) = \begin{cases} \prod(i,j), & if\ |i-j|a_c \geq 2r_{min} \\ \Omega(i,j), & else\ |i-j|a_c < 2r_{min} \end{cases}$$

The first two equations calculate the distance for the manoeuvres ∏ and Ω between tracks i and j. While last equation indicates, from the turning radius r_{min} of a vehicle and the track width a_c, when one turn or the other should be applied.

Once it is known how to find the minimum trajectories for the transitions, either by directly connecting the tracks or by interleaving a visit to the depot, it is sufficient to connect them with the intra-track trajectories to form the route for each vehicle.

Figure 56 shows, as an example, the trajectories encoded in the solution formed by the permutation $\sigma = (p_3, p_6, p_8, p_{10}, p_{10}, s_2, p_7, p_2, p_1, s_1, s_3, p_9, p_4, p_5)$ and the binary replenishment vector b = (1,0,1,0,0,0,0,0,0,0,0,0,0,0). The field is composed of 10 tracks ($|P|$ = 10), each in turn consisting of 4 crop rows. The field contains stands of weeds (green rectangles) that need to

be treated by the vehicles by spraying herbicide, for which they have 4 m spray booms with 4 nozzles, one for each tramline. The fleet has 4 vehicles, i.e., $|S| = 3$, although the third vehicle is not part of the solution because in σ there is no track in its charge. Initially it is known whether the vehicles start from the upper or lower headland, so it is easy to infer which end of track they have to start from. The turning width of the vehicles is 3 m, which allows transit between adjacent runways by Ω manoeuvres, as shown in the transition between the first and second runway.

The refuelling vector indicates that vehicles treating the first and third tracks of the field must go to the depot before starting to treat the following tracks. However, since the first track has no next track (it is the last track that vehicle 2 treats), in that case it is not necessary to go to refuelling.

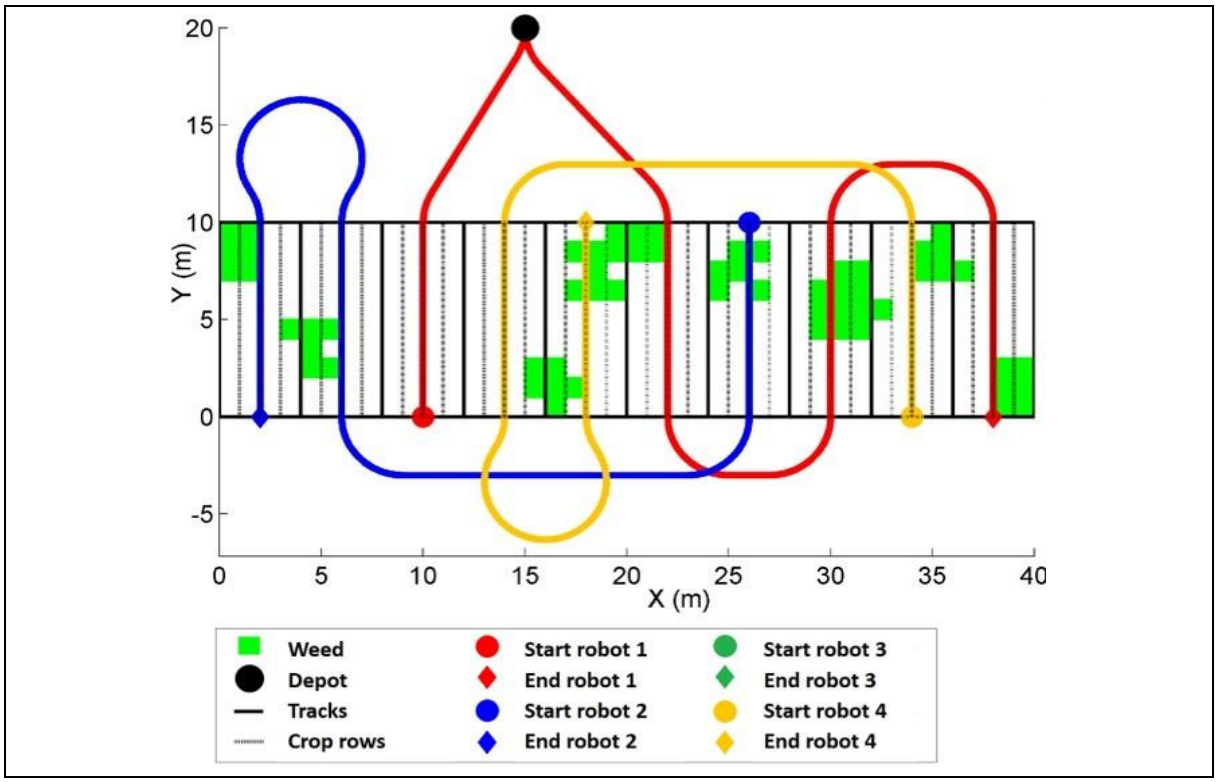

Figure 56 Trajectories associated with the permutation $\sigma = (p_3, p_6, p_8, p_{10}, p_{10}, s_2, p_7, p_2, p_1, s_1, s_3, p_9, p_4, p_5)$ and the vector of refuelling $b = (1,0,1,0,0,0,0,0,0,0)$

In addition to the trajectories, it is possible to automatically extract from the solution the action plan of the task to be performed, i.e., the status (speed, implement on or off) that the vehicles must have at each of the points along the routes. For this, it requires a list of the points and orientations defining the trajectories (entry and exit points and orientations of the tracks or the depot) together with their associated states, plus the intermediate points and orientations where there is a change in the state affecting the execution of the task. To calculate the status of the implement at each point, in addition to the trajectories, it is necessary to use additional information available from the field and the task, in this case the weed map indicating where the stands are. From the trajectories, the boom length, the

distribution of the nozzles on the boom, and the position of the stands it is easy to calculate at which points the nozzles should be opened and closed to spray only the weeds. Table 23 shows, as an example, the plan of vehicle 1 of the solution in Figure 56.

Vehicle	Step	Position	Orientation	Speed (km/h)	Nozzles of the Spray boom				Description
					V1	V2	V3	V4	
1	1	(10, 0)	90°	6	OFF	OFF	OFF	OFF	Starting point
1	2	(10, 10)	90°	10	OFF	OFF	OFF	OFF	Exit of track 3
1	3	(15, 20)	90°	10	OFF	OFF	OFF	OFF	Arrival at depot
1	4	(15, 20)	270°	10	OFF	OFF	OFF	OFF	Departure from depot
1	5	(22, 10)	270°	6	OFF	OFF	**ON**	**ON**	Start track 6
1	6	(22, 8)	270°	6	OFF	OFF	OFF	OFF	Change in nozzles
1	7	(22, 0)	270°	10	OFF	OFF	OFF	OFF	End of track 6
1	8	(30, 0)	90°	6	OFF	OFF	OFF	OFF	Start track 8
1	9	(30, 4)	90°	6	OFF	**ON**	**ON**	**ON**	Change in nozzles
1	10	(30, 7)	90°	6	OFF	OFF	**ON**	**ON**	Change in nozzles
1	11	(30, 8)	90°	6	OFF	OFF	OFF	OFF	Change in nozzles
1	12	(30, 10)	90°	10	OFF	OFF	OFF	OFF	End of track 8
1	13	(38, 10)	270°	6	OFF	OFF	OFF	OFF	Start track 10
1	14	(38, 8)	270°	6	OFF	OFF	OFF	**ON**	Change in nozzles
1	15	(38, 7)	270°	6	OFF	OFF	OFF	OFF	Change in nozzles
1	16	(38, 3)	270°	6	**ON**	**ON**	OFF	OFF	Change in nozzles
1	17	(38, 0)	270°	0	OFF	OFF	OFF	OFF	End point

Table 23 Plan of vehicle 1 coded in the solution in Figure 56

For each step, the vehicle has to adopt the associated state and maintain it until the next step. For example, in the steps related to the eighth track of the field (steps 8 to 12), it is indicated that the unit has to start from point (30, 0) with an orientation of 90° at a speed of 6 km/h. All boom nozzles must be switched off until the robot reaches point (30, 4) with the same speed and orientation, there the nozzles V2, V3 and V4 (the three rightmost nozzles) have to be activated to start spraying the stand in the middle of the track. Everything will remain the same until point (30, 7) where the V2 nozzle will be deactivated because the stand is reduced in width on the left and, one metre further on at point (30, 8), the other two valves, V3 and V4, will be deactivated. Finally, everything remains the same until the end of the runway, point (30, 10) where the speed is changed to 10 km per hour. Note that at the headland there are no requirements on the speed, so the robot can increase its speed to transit to the next track,

where, once again, the orders to open and close the nozzles to treat the weeds that appear on this track can be seen.

Finally, in this example it has been assumed that the starting positions of the vehicles were at the ends of their respective initial tracks. It is important to note that the procedure is the same if the starting points are outside the field, e.g., the location of a garage, a depot or any other point. In this case, it is only necessary to add to the vehicle trajectory, the initial transition that allows to go from the starting point with an initial orientation to the entry point on the first track with the required orientation. Similarly, when the end points are outside the crop, the final transitions from the exit points of the course to the defined end points shall be added to the trajectory.