BIM prove

D2.3 – Open Interface Description for stationery, ground and UAV based data acquisition systems

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement nº 958450. This document reflects only the author's view and the Commission is not responsible for any use that may be made of the information it contains.

Project Title	Improving Building Information Modelling by Realtime Tracing of Construction Processes
Project Acronym	BIMprove
Grant Agreement No	958450
Instrument	Research & Innovation Action
Topic	Industrial Sustainability
Start Date of Project	1st September 2020
Duration of Project	36 Months

Name and Number of the deliverable	2.3 - Open Interface Description for stationery, ground and UAV based data acquisition systems
Related WP number and name	WP 2 - Components, technologies & functionalities
Deliverable dissemination level	Public
Deliverable due date	30 September 2022
Deliverable submission date	30 September 2022
Task leader/Main author	Dag Fjeld Edvardsen (CATENDA)
Contributing partners	ZHAW, FhG-IAO, SINTEF, ROBI, VTT
Reviewer(s)	Andrej Cibicik (SINTEF)

Abstract

The deliverable describes how open software and hardware interface is defined in BIMprove and the implementation is described here for stationery and ground-based data acquisition systems. In most cases, the purpose of the data acquisition is to gather (3D point-cloud scans, photographs, other sensory input) observations for comparison of schedules, BIM-models and other expectations. This information is the main source for detecting mismatches and warnings from the BIMprove processes.

Keywords

Software development, user interfaces, UxV, IFC, data exchange

Revisions

Version	Submission date	Comments	Author
v0.1	15.08.2022	Initial version	Dag Fjeld Edvardsen (CATENDA)
v0.2	15.09.2022	Corrections	Matthias Aust (FhG-IAO), Kaj Helin (VTT), Ruprecht Altenburger (ZHAW)
V1.0	30.09.2022	Approved, final version	Andrej Cibicik (SINTEF)

Disclaimer

Acronyms and definitions

Acronym	Meaning
GS	Ground Station
BE	Backend
RPI4	Drone Computer (currently Raspberry PI 4)
FC	Flight Controller
DAC	Data Acquisition Computer
ROS	Robot Operating System
HMI	Human Machine Interface
BCF	BIM Collaboration Format
UAV	Unmanned Aerial Vehicle
UxV	Unmanned x Vehicle (x can be ground or aerial)
BIM	Building Information Modelling
GUI	Graphical User Interface
ROS	Robot Operative System

BIMprove project

In the past 20 years, productivity in the European construction industry has increased by 1% annually only, which is at the lower end compared to other industrial sectors. Consequently, the sector has to step up its digitization efforts significantly, on the one hand to increase its competitiveness and on the other hand to get rid of its image as dirty, dangerous and physical demanding working environment. Construction industry clearly needs to progress beyond Building Information Modelling when it comes to digitizing their processes in such a way that all stakeholders involved in the construction process can be involved.

The true potential of comprehensive digitization in construction can only be exploited if the current status of the construction work is digitally integrated in a common workflow. A Digital Twin provides construction companies with real-time data on the development of their assets, devices and products during creation and also enables predictions on workforce, material and costs.

BIMprove facilitates such a comprehensive end-to-end digital thread using autonomous tracking systems to continuously identify deviations and update the Digital Twin accordingly. In addition, locations of construction site personnel are tracked anonymously, so that **BIMprove** system services are able to optimize the allocation of resources, the flow of people and the safety of the employees. Information will be easily accessible for all user groups by providing personalized interfaces, such as wearable devices for alerts or VR visualizations for site managers. **BIMprove** is a cloud-based service-oriented system that has a multi-layered structure and enables extensions to be added at any time.

The main goals of **BIMprove** are a significant reduction in costs, better use of resources and fewer accidents on construction sites. By providing a complete digital workflow, BIMprove will help to sustainably improve the productivity and image of the European construction industry.

Contents

Index of Figures

Index of Tables

1. Introduction

BIMprove is a complex system, consisting of "satellites" working together. Between them - and sometimes inside them - there are interfaces defining how information is transferred. Some of this communication is real time (e.g., some of the information exchange related to the flight controller for the drone), other are large data transfers that happen after certain events (e.g., uploading of point-clouds to the central server – aka. “backend”). Many protocols and formats are involved, which will be described in this deliverable. One of the main novelties is that we use BCF as a central format for purposes, including the initiating request for scanning and the final decision at the end of the loop, which was not the original intended use of BCF. With such setup, we keep the overall BIMprove system standardized and show a possible way to use BCF in a new setting.

The following chapters highlights how some key data capture systems will communicate with the backend, how data acquisition is achieved from stationary, ground and UAV as the core concept in BIMprove.

In most cases the purpose is to gather (3D point-cloud scans, pictures, additional sensory inputs) observations for comparison of schedules, BIM-models and other expectations. We use this to make sure that we are on-track in the construction project: regarding safety, productivity and quality. In addition, we can also locate and visualize UxV positions in a BIM-context.

If there is a mismatch between BIM/schedule and the observation made by the data acquisition, we have to decide if it is important to change: rework might be needed or alternatively if rework is deemed too expensive the BIM-model should be changed if acceptable (to keep as-built).

We also gather observations to detect safety features and potentially warn if there are deviations. This can be used for safety analysis and early warning.

In addition, all this data capture contributes to the “ground-truth” part of the Digital Twin that is generated and kept up to date through the construction phase.

A key aspect in BIMprove is to use open standards as much as possible, and in cases where such does not exist, we use simple formats. This makes presenting this document significantly easier, since so many of the formats used are open and well documented standards. In some cases, we must introduce our own micro-formats, these are typically for transforming small pieces of data, where open formats do not exist, or where minimal data size is important for performance (like flight control). This following chapters will show these key interfaces/formats used for communication.

1.1. Well known open formats referred to in this document

- BCF: BIM Collaboration Format. buildingSMART documentation, https://technical.buildingsmart.org/standards/bcf/
- IFC: Industry Foundation Classes. buildingSMART documentation, https://www.buildingsmart.org/standards/bsi-standards/industry-foundation-classes/
- JPEG/EXIF: Open format for images with support for strong reduction in file size using lossy compression. Exif contains metadata Wikipedia information, https://en.wikipedia.org/wiki/JPEG
- E57. Popular open standard point-cloud format. Information on FileFormat.com, https://docs.fileformat.com/3d/e57/
- LAS: Another popular open standard point-cloud format. Wikipedia information, https://en.wikipedia.org/wiki/LAS_file_format
- JSON: JavaScript Object Notation. Information on Json.org, https://www.json.org/json-en.html
- CSV: Comma separated values. Information on Wikipedia, https://en.wikipedia.org/wiki/Comma-separated_values

The two last formats (JSON and CSV) are more like containers, they are used to compose small micro-formats.

2. The systems involved and key information flow

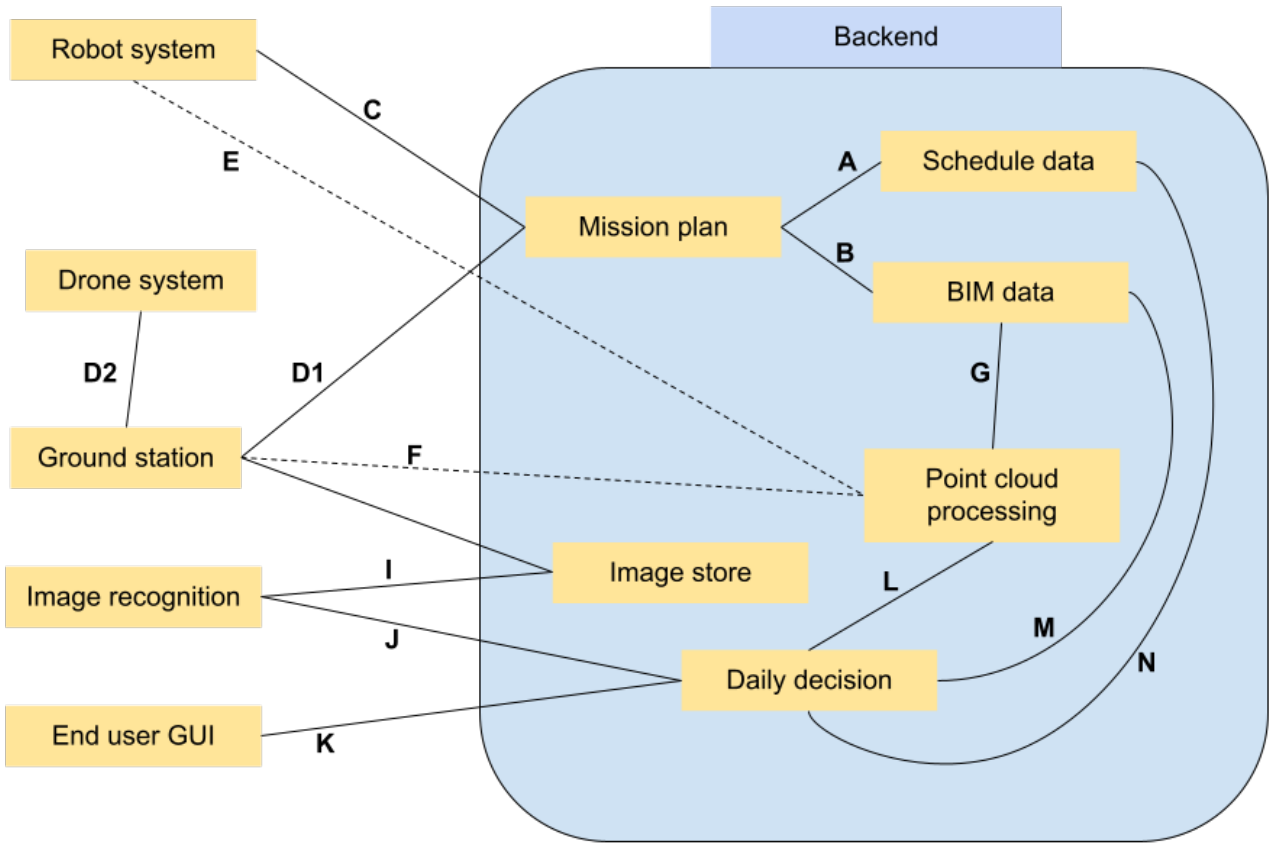

Figure 1 Schematic high-level visualization of the parts involved, and what their main communication is about

Figure 1 illustrates key information exchange related to UxV and ground unit data units. These are subset of the total BIMprove system since AR, VR, PCs, smartphones and tablets are not direct parts of this information flow. The Digital Twin and related statistical analysis is also excluded in this overview.

The information flow lines are labelled with letters indicating communication between two separate parts ("satellites") of the system. Each of these lines represent one or more information exchange, and the direction of arrowheads only indicate the initiating communication. In Table 1, a short description of each of these communication lines are listed, details are in the chapters later in the document.

Table 1 Description of communication lines, related to Figure 1

Line	Description of communication	Formats
A	Schedule data is pulled from the schedule system (with BIM references/links) into the mission high-	BCF issue

	level mission planning system. Based on this, a BCF issue is assigned to the UxV operator - this contains references to the BIM objects that the scan request is about.	
B	BIM data is pulled for the detailed mission planning - when creating a set of way points and scan points for the mission, the BIM model provide useful information for navigation planning.	Geometry data requested from model server directly
C	The mission plan is sent to the robot.	BCF issue + JSON
D1	The mission plan is sent to the drone.	BCF issue + JSON
D2	Mission data and real-time data between ground station and drone.	Drone data link protocol and file transfer of images
E	After the physical 3D scanning mission is complete, the BIM-aligned point cloud is sent from the robot system to the processing part of the backend.	• Direct file transfer robot to < cyclone software • REST message with JSON for robot position
F	After the physical 3D scanning mission is complete, the BIM-aligned point cloud is sent from the drone system to the processing part of the backend.	• Direct file transfer RPI4 → GS → Pix4D • REST message with JSON for drone position
G	BIM data is sent to the processing part of the backend (to compare with the observed point-clouds)	• REST based point cloud upload, format is Las or e57
H	Images (aligned as a bonus effect of being used for photogrammetry) is sent to the image store in the backend.	• REST based upload of images, format is JPEG with embedded EXIF information

I	The aligned images are sent to the image recognition service.	• REST based upload of images, format is jpeg with embedded EXIF information
J	The recognized images are sent to the backend, where those who should have seen safety features (but did not) are sent to the daily decision cycle.	• BCF issue with attached annotated message and embedded JSON information
L	The processed result of point-clouds (3D scan) with BIM (plan) is sent to the daily decision cycle. Each BIM-object asked for get a separate BCF issue, with estimated of position deviation (x,y,z) provided, together with a colour coded (heatmap) local point-cloud. This is presented to the user with the original BIM models for context.	• BCF issue with linked point cloud and BIM objects
M	Based on the result of daily decisions, significant deviations can lead to updating the BIM model so that it stays close to as-built for future Facilities Management purposes.	• IFC
N	Based on the result of daily decisions, if a significant rework is needed the schedule is updated to allow this. Also, the status of progress and cost us updated if they have changed.	• Microsoft Project (MPP) or IFC or direct task change via web GUI

According to the requirements stated in D1.2 Requirements list of BIMprove, the BIMprove system requires a set of BIM-models and a construction schedule. These will be updated through the construction project, resulting in a Digital Twin for the construction process, and make an important input for initial step in each daily cycle - the high-level scan request.

2.1. High-level scan request

The scan request from the BIM-operator (“high-level scan request”) is in the form of a BCF-issue. This is then assigned to an UxV operator: the components field of a BCF viewpoint includes a list of

BIM-components (via IFC GUIDs) that the BIM-operator wants to be scanned. In this way the scan request is easily readable by both a machine and a human (the latter via standard software, or a text editor if needed).

Formats and endpoints (listed together for clarity):

- BCF

  Bimsync is used as the BCF server. The endpoints that will be used are:

  - Create topic (an example for shortness, the schema is longer in order to keep example is as clear as possible)

```
POST https://opencde.bimsync.com/bcf/2.1/projects/c0784b27-5057-43f6-bdb3-3607f4da86b0/topics
{
   "topic_type": "Scan",
   "topic_status": "Not yet accepted",
   "title": "Scan new columnns on floor 4",
   "labels": [
      "label1"
   ],
   "creation_date": "2022-10-03T08:06:39.264+0000",
   "creation_author": "king@test.com",
   "bimsync_assigned_to": {
      "user": {
         "ref": "e4d94d45f7194f1f9f7b29ed994d01e4",
         "email": "king@test.com",
         "name": "King"
      },
      "team": "drone_team"
   }
}
```

  - Create viewpoint

```
POST https://opencde.bimsync.com/bcf/2.1/projects/3c34c9b3-1b9b-4750-a4f3-0641d58fe48e/topics/6411ce04-5391-40a6-97c2-be0ca45fcc96/viewpoints
{
   "index": 2,
   "components": {
      "selection": [{
         "ifc_guid": "3BY5alwwL00QXHtY2qmHYZ",
         "authoring_tool_id": "BIMprove_High_Level_mision_requester"
      }],
      "coloring": [{
         "color": "#dddddd",
         "components": [{
            "ifc_guid": "3BY5alwwL00QXHtY2qmHYZ"
         }]
      }],
      "visibility": {
         "default_visibility": true,
         "exceptions": [{
```

```
          "ifc_guid": "0pQa7h_GLBZfPdI0XNo8Ru",
          "authoring_tool_id": "BIMprove_High_Level_mision_requester"
        }],
        "view_setup_hints": {
          "spaces_visible": true,
          "space_boundaries_visible": true,
          "openings_visible": true
        }
      }
    },
    "perspective_camera": {
      "camera_view_point": {
        "x": -1.3757544290857453,
        "y": -14.084548687547457,
        "z": 15.77535220227417
      },
      "camera_direction": {
        "x": 0.48836276817323054,
        "y": 0.6442499485024906,
        "z": -0.5885947761547989
      },
      "camera_up_vector": {
        "x": 0.3555637551166773,
        "y": 0.4690610051624899,
        "z": 0.808428221602439
      },
      "field_of_view": 60.0
    },
    "lines": [],
    "clipping_planes": [],
    "snapshot": {
      "snapshot_data":
"R0lGODlhPQBEAPeoAJosM//AwO/AwHVYZ/z595kzAP/s7P+goOXMv8+fhw/v739/f+
8PD98fH/8mJl+fn/9ZWb8/PzWlwv///6wWGbImAPgTEMImIN9gUFCEm/gDALULDN8
PAD6atYdCTX9gUNKlj8wZAKUsAOzZz+UMAOsJAP/Z2ccMDA8PD/95eX5NWvsJC
OVNQPtfX/8zM8+QePLl38MGBr8JCP+zs9myn/8GBqwpAP/GxgwJCPny78lzYLgjAJ
8vAP9fX/+MjMUcAN8zM/9wcM8ZGcATEL+QePdZWf/29uc/P9cmJu9MTDImIN+/r7+/
vz8/P8VNQGNugV8AAF9fX8swMNgTAFlDOICAgPNSUnNWSMQ5MBAQEJE3QPIG
AM9AQMqGcG9vb6MhJsEdGM8vLx8fH98AANIWAMuQeL8fABkTEPPQ0OM5OSYd
GFl5jo+Pj/+pqcsTE78wMFNGQLYmID4dGPvd3UBAQJmTkP+8vH9QUK+vr8ZWSH
pzcJMmILdwcLOGcHRQUHxwcK9PT9DQ0O/v70w5MLypoG8wKOuwsP/g4P/Q0Icw
KEswKMl8aJ9fX2xjdOtGRs/Pz+Dg4GImIP8gIH0sKEAwKKmTiKZ8aB/f39Wsl+LFt8d
gUE9PT5x5aHBwcP+AgP+WltdgYMyZfyywz78AAAAAAD///8AAP9mZv///wAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAACH5BAEAAKgALAAAAAA9AEQAAAj/AFEJHEiwoMGDCBMqXMiwocAbBw
w4nEhxoYkUpzJGrMixogkfGUNqlNixJEIDB0SqHGmyJSojM1bKZOmyop0gM3Oe2liT
ISKMOoPy7GnwY9CjIYcSRYm0aVKSLmE6nfq05QycVLPuhDrxBlCtYJUqNAq2bNW
EBj6ZXRuyxZyDRtqwnXvkhACDV+euTeJm1Ki7A73qNWtFiF+/gA95Gly2CJLDhwEH
MOUAAuOpLYDEgBxZ4GRTlC1fDnpkM+fOqD6DDj1aZpITp0dtGCDhr+fVuCu3zlg49
```

ijaokTZTo27uG7Gjn2P+hI8+PDPERoUB318bWbfAJ5sUNFcuGRTYUqV/3ogfXp1rWl
Mc6awJjiAAd2fm4ogXjz56aypOoIde4OE5u/F9x199dlXnnGiHZWEYbGpsAEA3QXYn
HwEFliKAgswgJ8LPeiUXGwedCAKABACCN+EA1pYIIYaFlcDhytd51sGAJbo3onOpa
jiihlO92KHGaUXGwWjUBChjSPiWJuOO/LYIm4v1tXfE6J4gCSJEZ7YgRYUNrkji9P55
sF/ogxw5ZkSqIDaZBV6aSGYq/IGZplndkckZ98xoICbTcIJGQAZcNmdmUc210hs35n
CyJ58fgmIKX5RQGOZowxaZwYA+JaoKQwswGijBV4C6SiTUmpphMspJx9unX4Kai
mjDv9aaXOEBteBqmuuxgEHoLX6Kqx+yXqqBANsgCtit4FWQAEkrNbpq7HSOmtwag
5w57GrmlJBASEU18ADjUYb3ADTinIttsgSB1oJFfA63bduimuqKB1keqwUhoCSK374
wbujvOSu4QG6UvxBRydcpKsav++Ca6G8A6Pr1x2kVMyHwsVxUALDq/krnrhPSOzX
G1lUTIoffqGR7Goi2MAxbv6O2kEG56I7CSlRsEFKFVyovDJoIRTg7sugNRDGqCJzJg
cKE0ywc0ELm6KBCCJo8DIPFeCWNGcyqNFE06ToAfV0HBRgxsvLThHn1oddQMrX
j5DyAQgjEHSAJMWZwS3HPxT/QMbabI/iBCliMLEJKX2EEkomBAUCxRi42VDADxyT
YDVogV+wSChqmKxEKCDAYFDFj4OmwbY7bDGdBhtrnTQYOigeChUmc1K3QTnA
UfEgGFgAWt88hKA6aCRIXhxnQ1yg3BCayK44EWdkUQcBByEQChFXfCB776aQsG
0BllQgQgE8qO26X1h8cEUep8ngRBnOy74E9QgRgEAC8SvOfQkh7FDBDmS43Pm
GoliKUUEGkMEC/PJHgxw0xH74yx/3XnaYRJgMB8obxQW6kL9QYEJ0FIFgByfIL7/I
QAlvQwEpnAC7DtLNJCKUoO/w45c44GwCXiAFB/OXAATQryUxdN4LfFiwgjCNYg+k
YMIEFkCKDs6PKAIJouyGWMS1FSKJOMRB/BoIxYJIUXFUxNwoIkEKPAgCBZSQH
Q1A2EWDfDEUVLyADj5AChSIQW6gu10bE/JG2VnCZGfo4R4d0sdQoBAHhPjhIB94
v/wRoRKQWGRHgrhGSQJxCS+0pCZbEhAAOw==" }
}

- Get selection (the IFC GUIDs in connected to the issues - used for high-level scan request with indication of target objects)

GET https://opencde.bimsync.com/bcf/2.1/projects/3c34c9b3-1b9b-4750-a4f3-0641d58fe48e/topics/6411ce04-5391-40a6-97c2-be0ca45fcc96/viewpoints/747f2fe6-b690-471f-8e45-618a4bc9a63c/selection

This can then return:

"selection": [
{
"ifc_guid": "3BY5alwwL00QXHtY2qmHYZ",
"authoring_tool_id": "BIMprove_High_Level_mision_requester"
}
]
}

- IFC (open standard for building information models)

3. Technical mission plan (general)

This high-level scan request is received by the UxV operator, who using a technical mission planning tool, which indicates start position, final landing position (typically the same), and waypoints and scan-points. These are stored in a simple JSON-format, and are attached to the high-level scan request BCF issue. The waypoints are hints for how to reach the scan-points, but the UxVs have autonomous capabilities and can find paths around obstacles. Using the control station, the UxV operator uploads the technical mission plan to the UxV unit. Geometry from IFC together with the schedule is used to estimate, which parts of the buildings are already built or in progress - relevant information to plan a route around obstacles.

Formats:

- BCF
- IFC
- JSON for sending technical mission plan from the planning tool to the UxV control station:

```
[{
        "kind": "start_position",
        "x": 0.0,
        "y": 0.0
    },
    {
        "kind": "waypoint",
        "x": 1.1,
        "y": 2.2
    },
    {
        "kind": "scanpoint",
        "x": 3.3,
        "y": 4.4,
        "theta": 1.5
    },
    {
        "kind": "landing_point",
        "x": 5.5,
        "y": 6.6
    }
]
```

This is being used as input for further planning and executing by the drone or the robot, as will be shown in the next chapters.

4. Data communication between drone – ground station and backend

This chapter describes the data communication regarding the drone data capture. Figure 2. gives a schematic overview of the different included devices.

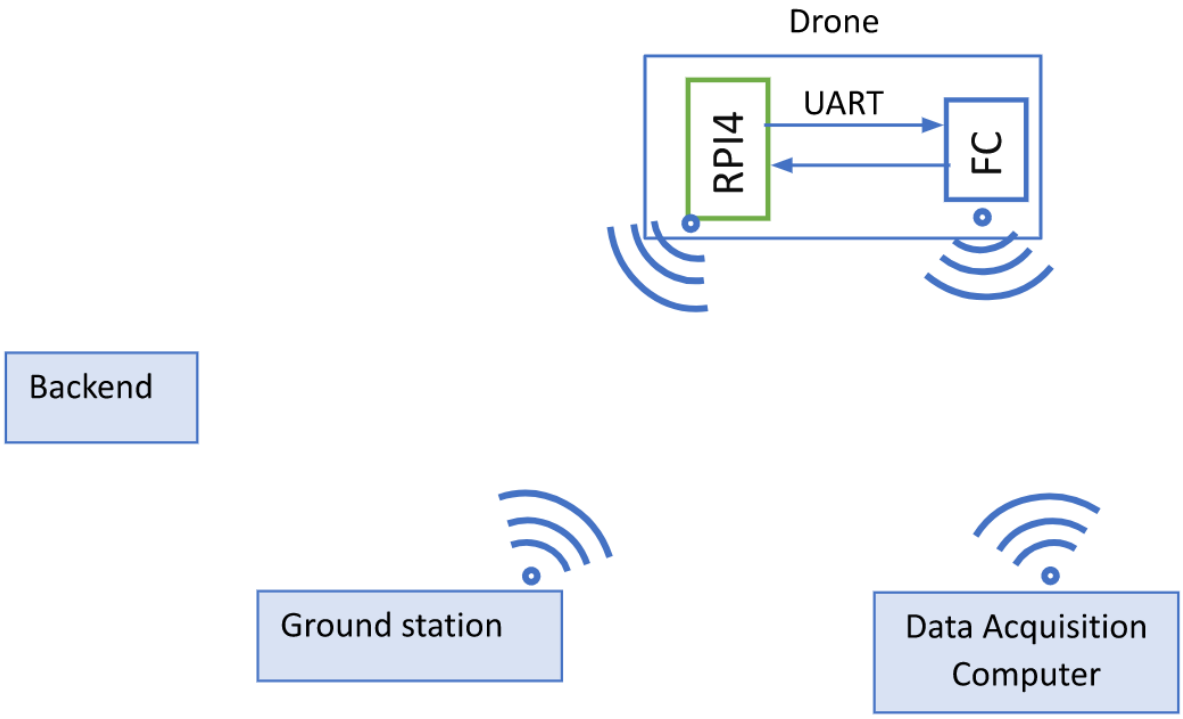

Figure 2 Schematic overview of the different devices for drone data capture

4.1. On the Drone

The main controller units on the drone are the “Drone Computer” currently being a Raspberry PI4 with 8 GB RAM. This device does on-board path planning, communication with the ground station, but also data connection to the Intel RealSense depth camera and the RGB camera (as described in D3.3 Proof of Concept System Description and Test Results). These connections to the cameras are not discussed in detail, for more information regarding that, see: [2], [3]. Furthermore, the Drone Computer is connected to the flight controller, which is currently an ARM H7 from STMicroelectronics. This is the core system for stabilizing the platform and is connected to sensors and devices (IMU, Lidar, Barometer, Gimbal), but not discussed in detail here. Communication with these devices is with the typical busses like SPI and I2C.

4.2. Ground Station

The ground station computer is typically in close proximity to the area that has to be scanned (a specific point on a floor on site). It has a wired Ethernet connection to the backend and a local WiFi

to the Drone Computer. The ground station is the connection node from the drone to the backend. It has a “Drone Cockpit” software that helps planning and starting autonomous flights etc., as presented in D3.2 Functional samples description and test results.

Figure 2. also shows the Data Acquisition Computer which is connected via WiFi directly to the flight controller. This connection is just for development purposes and is not discussed in detail here.

4.2.1. Data capture with drones

There are different data streams between drones and the backend with sometimes very different requirements in terms of data volume and real-time capability. Table 2 lists typical data packages transferred between different devices:

Table 2 list of various data types exchanged

What	Sender	Receiver	Typical data vol.	Realtime	During flight
Request for mission	BE	GS	100 kByte	No	No
Mission Task	GS	RPI4	5 k Byte	No	No
Desired Drone Positions	RPI4	FC	100 Byte	Yes	Yes
Flight state	FC	RPI4	100 Byte	Yes	Yes
Flight state	RPI4	GS	100 Byte	Med	Yes
Flight state	GS	BE	1 kByte	No	Yes
Image Data	RPI4	GS	1 GByte	No	No
Image Data	GS	BE	1 GByte	No	No

4.3. Example data transfer of captured images

During a typical mission typically around 100-200 high resolution images are captured with the on-board camera. This data is not transferred during flight, a temporal TCP-IP connection is only set up after landing to transfer the files.

Example transfer of the flight state (see Table 2)

The flight state describes the most important states of the drone, like its' position and orientation in space, if it is flying or charging etc. Most states are captured and calculated on the FC on the drone. Since this data is always needed in real-time on the RPI4 to do its path-planning tasks, there is a constant bidirectional data connection between them running at 50Hz. The data protocol is defined in detail further down. The data might also be needed for checks on the GS and the BE for visualization purposes and checks. This is done at a lower priority and frequency.

4.4. Two lines of data transfer

For the transfer of captured photos, a TCP-IP protocol is chosen because of a guarantee of the data correctness at the receiving end. Since this occurs after landing, a WiFi connection between RPI4 and GS of good quality should be easily possible. The RPI4 will be acting as a client while the GS is performing as a server. However, the initial communication will be initialized and done through the pre-defined UDP protocol for prompting both ends for the transfer tasks.

All other data transfer consists of small data packages and partly of strong real time requirements. The connection can be wired (FC – RPI4, GS – BE) or wireless (RPI4 – GS, FC – DAC)

4.5. Drone Data Link protocol (DDL)

In drone control and data connection to ground stations there is a well-established data protocol called “MavLink” developed at ETH-Zurich from around 10 years ago [1]. This protocol is tailored for drone – ground communication, drone – sensor communication etc. It is a lean and well documented software that is under constant development in an open-standard manner. It was studied in detail, but it was decided not to include it, since the needs we have in BIMprove would need constant adaptions with data streams that are not typical for standard drone flights (though it would be possible in principle, using free data formats MavLink can work with).

A lean data protocol was developed which covers the needs we have for autonomous flights. It always consists of:

7 Byte header || *N-Bytes of data* (up to 2^{16}) || a *checksum*

Figure 3 describes the protocol bytewise in detail:

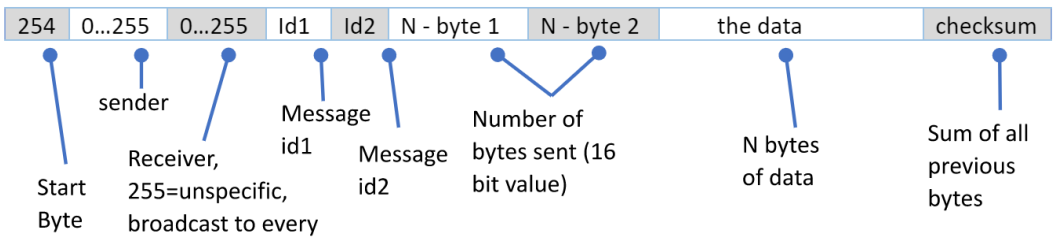

Figure 3 DDL byte map

Note that the numbers for “sender” and “receiver” are not needed in all cases, since sometimes a bi-directional wired connection between these partners is already established.

Generally, the broadcasted data can be of any type. What kind of data (vector of single precision values, ASCII-text, single bytes,…) is specified in the two message ids Id1 and Id2. This protocol is used for all above mentioned data exchange except the transfer of large files via TCP-IP. The actual transfer type can be UDP (the WiFi connections) or UART (RPI4 – FC).

Figure 4 shows a summary of definitions of id1 and id2. It must be noted that not all types for transfer are needed between all partners. For example, orientations and estimated positions (Id1 = 101, Id2 = 1,2) are sent as small packages between FC and RPI4 and high rates, whereas these values are packed to a larger data package and sent to the GS at lower rates (Id1 = 80, Id2 = 100).

id1	id2		nb bytes	datatype	FC->RPI4	RPI4->GS	GS->RPI4	RPI4->FC	FC->RPI4
101		EKF							
	1	estimated RPY values from EKF	3	float	x				x
	2	estimated xyz from EKF	3	float	x				x
230		Takeover, flightstate etc.							
	1	Acknoledge (dat = 1) or decline (dat =0) takeover by gui	1	uint8_t	x				x
	2	Motors are spinning!!	1	uint8_t					x
	10	sm_FS Flight state (ARMING, FLIGHT, WAIT_FOR...)	1	uint8_t	x				
	101	Request takeover by gui	1	uint8_t			x		
	102	Arm drone by gui	1	uint8_t			x	x	
	111	give back control to FC	1	uint8_t				x	
80	100	FC data package, consisting of:							
	"	x,y,z,R,P,Y,vx,vy,vz,Fthrust,Lidar, BaroAlt	12*4	float					x
	"	flight mode, flightState	2	uint8					x
			(50 in total)						
70		Filetransfer TCP							
	35	Request File from Drone	50	char[50], relative path		x	x		
	36	Incoming File to Drone	50	char[50], relative path		x	x		

Figure 4 Byte character interpretation between FC, RPI4 and GS

This protocol leads to very short and fast data transfers, such that for example the flight controller’s computational power is minimally affected and it can do its major tasks like calculating flight states and controller outputs, without interruptions.

Figure 5 shows a listing of data send function on the flight controller. This is called every 20ms.

```
/* send N/4 float values via UART */
void rpi4_comm_thread::send_f_data(char id1,char id2,uint16_t N,float *dat)
{
    buffer[0] = 254;                // startbyte
    buffer[1] = DRONE_ID;       // sent from this drone
    buffer[2] = 255;                // broadcast
    buffer[3] = id1;            // id1
    buffer[4] = id2;            // id2
    buffer[5] = N%256;          // high Byte
    buffer[6] = N/256;          // low Byte
    uart->write(buffer, LENGTH_HEADER);
    uint8_t *float_data = (uint8_t *)dat;
    uart->write(float_data,N);
    uint8_t csm = calc_csm(buffer, N + LENGTH_HEADER);
    uart->write(&csm,1);
}
```

Figure 5 code snippet from FC implementation

5. Unmanned Ground Vehicle - the robot

On the robot side a conversion between JSON and the ROS message format is needed to receive the waypoints for the scanning.

ROS [4] communication is via XML-RPC, which is a HTTP-based protocol. Inside the ROS system there are several nodes in charge of processing different operations (wheels control, laser communication, localization…) and for the communication between them uses Topics based on TCPROS [5] or UDPROS [6] protocols. These Topics use ROS messages which are data structures composed of different typed fields (as shown on Figure 6).

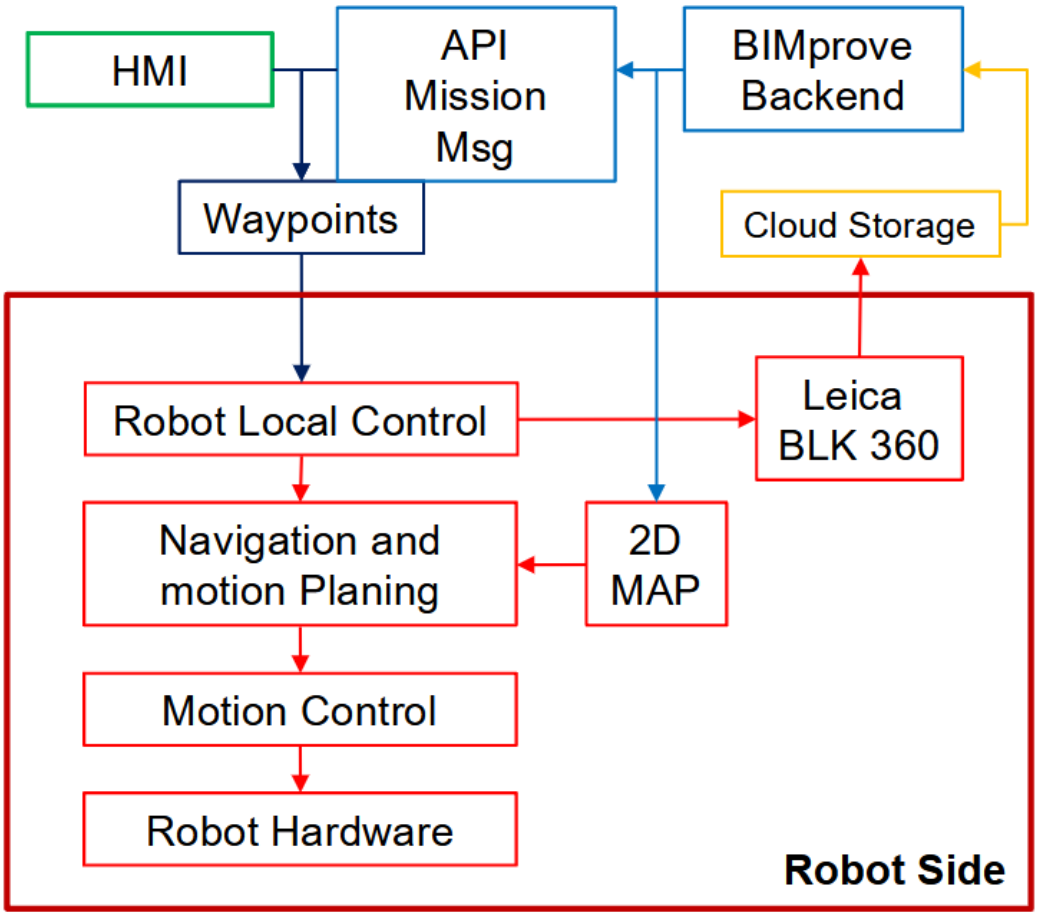

Figure 6 The UGV data transfers and its' connections

To interpret the missions from BIMprove backend in ROS based system, a ROS node was created for getting the JSON, then processing the sequence of waypoints for the scanning and adding this sequence to the robot sequencer manager. The sequences are saved in a yaml file with the name of the sequence as a header and the actions that will be done.

Here is an example of a sequence with 2 waypoints and the scanning command:

The waypoint has x, y coordinates from the map and the rotation wanted.

```
TEST_SEQUENCE:
- RLC_GOTO -1.42 -1.79 -1.5181
- BLK_DOWNLOAD_PC_ON_THE_FLY
- RLC_GOTO 3.53 -2.24 -1.50
- BLK_DOWNLOAD_PC_ON_THE_FLY
```

Once this process is done, the sequence will be available from Robotnik's HMI (shown in Figure 7) and the UGV operator will be able to preview, edit and launch the sequence (as shown on Figure 8) and also the location of the waypoints will be represented on the 2D map used for the robot for localization and navigation (discussed in detail in D3.3 Proof of Concept System Description and Test Results).

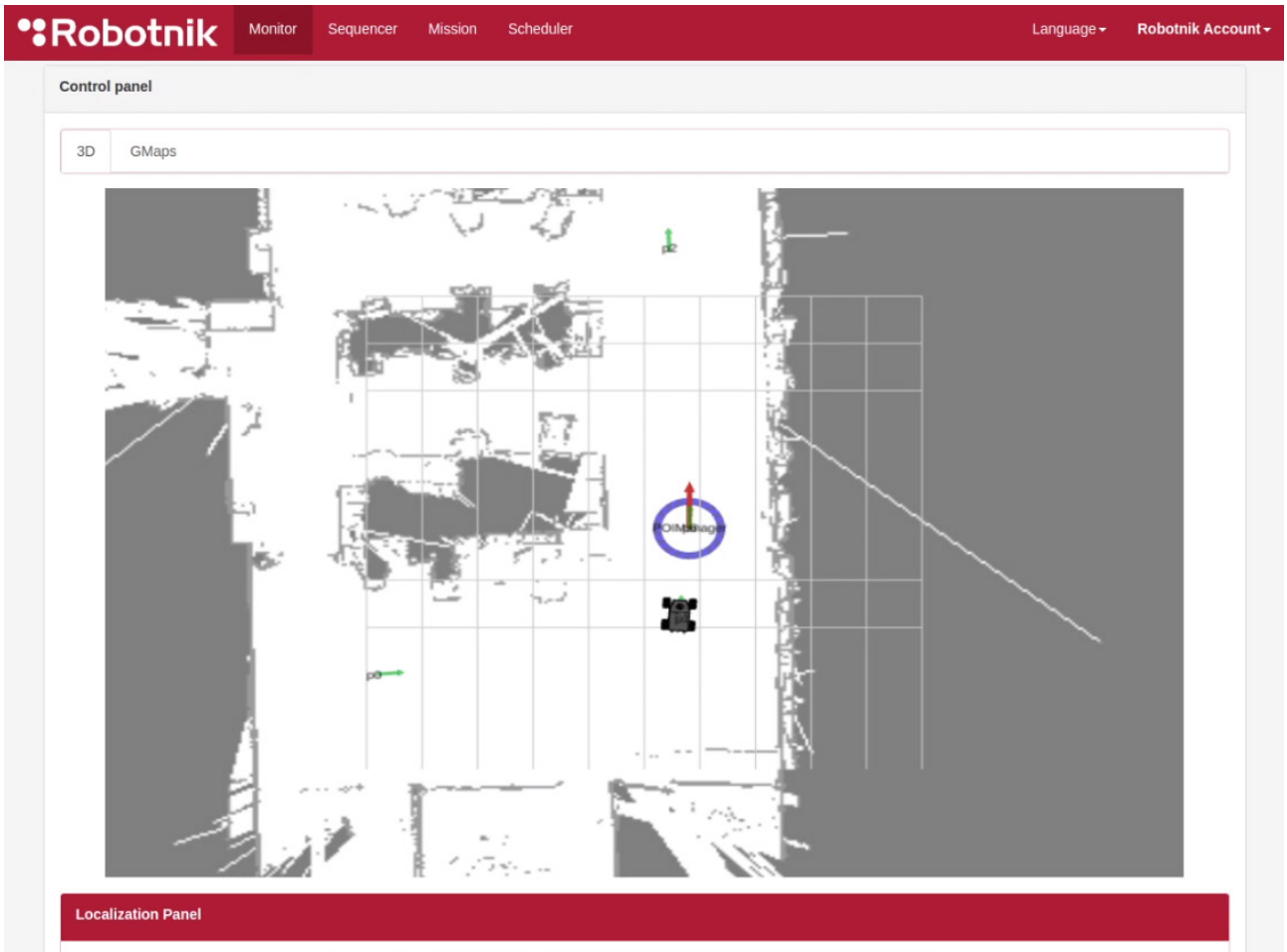

Figure 7 HMI - visualization of map and waypoints

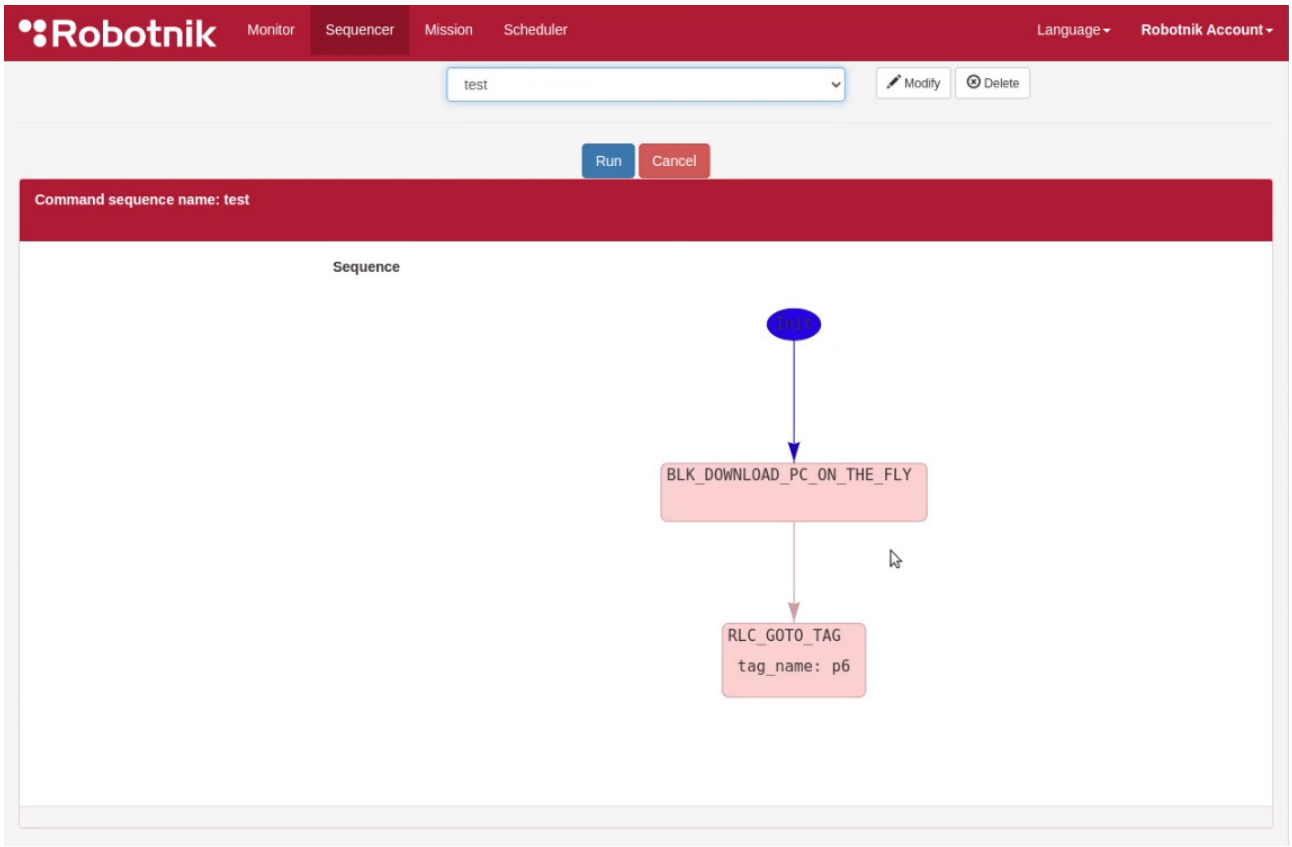

Figure 8 HMI - example sequence

6. Unmanned Ground Vehicle - data capture

The capture of data can be done with the BLK360 or the BLKARC laser scanners from Leica (shown on Figure 9 and 10). The scanners interfaces were modified in order to be compatible with ROS. A ROS node was made to add functions for start/stop the capture, list all the captures that are already done and download the finished point-clouds. With these functions available, the sequence manager can start and stop the scanning directly from ROS. The time needed to execute a capture (3D scanning) is depending on the precision that is requested and on the type of laser scanner.

In the case of the BLK360, this needs to stay stationary on the waypoint while is scanning and for a medium precision with colour will take around 2:30 minutes. Otherwise, the BLKARC scanning process can be done while the robot is movement but will need to pass more than one time by each waypoint to obtain enough points for the point-cloud.

After the scanning sequence is done, the point-clouds needs to be aligned with the BIM model and then uploaded to the cloud, at this moment the final process is done manually by the UGV operator.

Figure 9 RB-Summit with BLKARC laser on top

Figure 10 the BLK360 laser scanner - mounted on top of the robot

6.1. UxV position data (general)

When the UxV is carrying out its' scanning mission, it can send its' position/pose to the backend, so that the current position can be visualized in a web interface.

Formats:

- JSON schema defined in BIMprove (simple JSON object with device-id, mission-id, timestamp, position x, y, z, orientation).. coordinates in BIM-coordinate.

```
{
  "type": "object",
  "properties": {
    "zone_id": {
      "type": "string"
    },
    "device_id": {
      "type": "integer"
    },
    "timestamp": {
      "type": "string"
    }
  },
  "required": [
    "mission_id",
    "device_id",
```

```
    "timestamp"
  ]
}
```

6.2. UXV images and point cloud data - upload to the backend

When the UAV (flying drone) has finished its' scanning mission, it lands, and the photos are uploaded to the control station. The photogrammetry application Pix4D is used to generate a point-cloud and align its coordinate system with the BIM coordinate system.

When the UGV (robot) has finished its' scanning, the point-cloud is aligned in a point cloud application and then sent to the backend. To help with aligning the data, a set of markers with precise locations (identified by surveying professionals) are used. These markers have both chili-tag/april-tags and QR-codes (as described in D1.2 Requirements list of BIMprove). A CSV-file with 4 columns keeps track of this (columns are marker-id, x- ,y- , and z-position).

In both cases the point cloud (e57 or Las) is uploaded to the backend using a dedicated endpoint for each construction project. The filename contains the following metadata: Project-number, scanner-id, timestamp and (if available) BCF-issue-id for the original high-level scan request. An example of such a filename is: “P01_S02_2022_09_25_21_01_02_BCF_6c6cc9ee-4a16-4eb7-b875-e3b43d09d74d.e57”.

The action of uploading it to this specific URL with this specific filename triggers processing actions on the backend, including comparing the geometry of BIM-objects with point cloud at the same location. This last part is outside the scope of this document. This is also the case for the BCF-files containing information so that the decision maker can make judgement calls in the daily decision cycle (available at 07:00 each morning after data capture).

Formats:

- e57 or Las (see above)
- BCF (see above)
- CSV (simple list of marker positions, all numbers in same coordinate system as the BIM and in meters, shown in Table 3). This CSV file is uploaded to the backend, and each physical marker has a QR code that contains a string representing the URL of an endpoint. Using this, a person or machine can scan the QR code and get back the id and position of the marker.

Table 3 sample CSV file

id	x	y	z
101	100.5	20.3	18.4
102	110.2	15.4	18.4
103	97.3	23.7	21.4

Endpoints:

- Ask for marker position by id:
  - Ideally, the markers have been placed and their positions read via an endpoint. Since the dot-count increases with the length of the string (URL), the point size and readability of the QR code decreases. For that purpose the BIMprove backend presents a simplified URL for getting positions by sending in a URL following this example: cloud.bimprove-h2020.eu/m/1/101. The /m/1/101 part means (in order) m for the marker endpoint, 1 for the project mapped to this number, and 101 for the individual marker id. calling this endpoint returns id, x, y and z.
- Upload point cloud file to backend. The endpoint is at https://cloud.bimprove-h2020.eu/projects/{project-guid}/pc_upload. The file uploaded will have metadata in its filename (see above) so that the backend knows how to deal with it (processing based on BCF-components, further uploading to Bimsync for visualization etc).

6.3. Machine learning and image classification – data processing

Images are collected by the UAV for photogrammetry and the main purpose is to create point clouds. The software that does this as a result also knows the position of the camera for each of the photos involved. This information (6 degrees of freedom - position and pose) is saved in the EXIF format so that it is directly attached to each image. At the same time, we can define 3D volumes in the BIM coordinate system to indicate where there is a required to be safety nets and safety barriers. This is generated based on a "safety-model.json" file indicating in 2.5D (x,y and floor with elevation) where the safety nets/barrier volumes. This JSON file is converted to IFC and each entity is represented there by a IfcSpatialZone (an IFC4 feature).

By combining the images (with camera position/pose) and the safety model, we can identify the pictures where a safety net or safety fence is expected to be seen (given line-of-sight). Machine

learning with labelled images has been used to train an AI system so that it can recognize image with safety nets/barriers. This means that images where safety nets/barriers are expected to be seen, can be run through this AI to indicate if they actually show that. Using this process, we can identify which of the required safety nets/barriers that were looked at by a camera (again assuming line-of-sight) but where no such feature was indicated. This can then be used to indicate in the 3D model which safety net/fence model that has been looked at, but where no indication of it was found. Please see the D2.4 Detailed description of the safety functionalities and worker notifications, for more details.

Formats:

- Jpeg including Exif field with position and pose of the camera (estimated by Photogrammetry).
  Embedded Exif field (comment): {‘imageID’: ‘1661497983_t_DJI_0095.JPG_817bf103-ecbe-4400-890e-9405786948e0’, ‘name’: ‘barrier’, ‘confidence’: ‘0.265504092’, ‘xmax’: ‘4486.267578125’, ‘xmin’: ‘64.4344558716’, ‘ymax’: ‘3160.9418945312’, ‘ymin’: ‘2683.8967285156’, ‘localOrigFileName’: ‘1661497983_t_DJI_0095.JPG’, ‘localAnchorFileName’: ‘1661497983_bbxes_t_DJI_0095.JPG’, ‘imageURL’: ‘https://fasolt4.willab.fi:8883/public/31ae3385927b’, ‘anchorBoxImageURL’: ‘https://fasolt4.willab.fi:8883/public/2c5ef7e43b61’}
- Safety model indicating where safety nets/barriers should be (JSON with a set of typed 2D-polygons per floor + floor elevation and height). The format is designed to favour simplicity over generality. An example:

```
{
  "scheme": "BIMprove_safety_model",
  "building_name": "building_D",
  "storeys":
  [
    {
      "storey_name": "2nd floor",
      "z_level_meters": 4.75,
      "height": 3.0,
      "zones":
      [
        {
          "name": "safety fence 2nd SW",
          "kind": "barrier",
              "height": 1.2,
          "polygon":
          [
            {
              "x": -6.278,
              "y": -35.346
            },
```

```
            {
              "x": -12.09,
              "y": -28.537
            },
            {
              "x": -17.612,
              "y": -23.032
            },
            {
              "x": -24.254,
              "y": -18.247
            },
            {
              "x": -29.210,
              "y": -15.484
            },
            {
              "x": -34.148,
              "y": -12.812
            }
          ]
        }
      ]
    }
  ]
}
```


6.3.1. Image recognition endpoint

Below is a Python example of sending a picture to the image recognition service (further detailed in D2.5 Safety-critical situation detection):

```
import pprint
import requests
DETECTION_URL = http://<SERVICE_URL>/v1/risk_objects/
TEST_IMAGE = "20.jpg"
image_data = open(TEST_IMAGE, "rb").read()
response = requests.post(DETECTION_URL, files={"image": image_data}).json()
pprint.pprint(response)
```

Answer from the endpoint can then be according to the following example


```
[{'class': 1,
  'confidence': 0.414974153,
  'name': 'safety_net',
  'xmax': 1620.0979003906,
  'xmin': 0.0,
  'ymax': 1082.6301269531,
  'ymin': 37.0090942383},
{'class': 1,
  'confidence': 0.3516817391,
  'name': 'safety_net',
```

```
  'xmax': 1724.8165283203,
  'xmin': 810.5069580078,
  'ymax': 610.0239257812,
  'ymin': 4.0267028809},
 {'class': 1,
  'confidence': 0.263441056,
  'name': 'safety_net',
  'xmax': 1870.6253662109,
  'xmin': 507.2440185547,
  'ymax': 1088.0,
  'ymin': 525.5383300781
 }]
```


This describes, where in the image a known object type is indicated to be (rectangle) and with which certainty it was identified.

Visually this looks like shown in Figure 11 (rectangle rendered by the image recognition system).

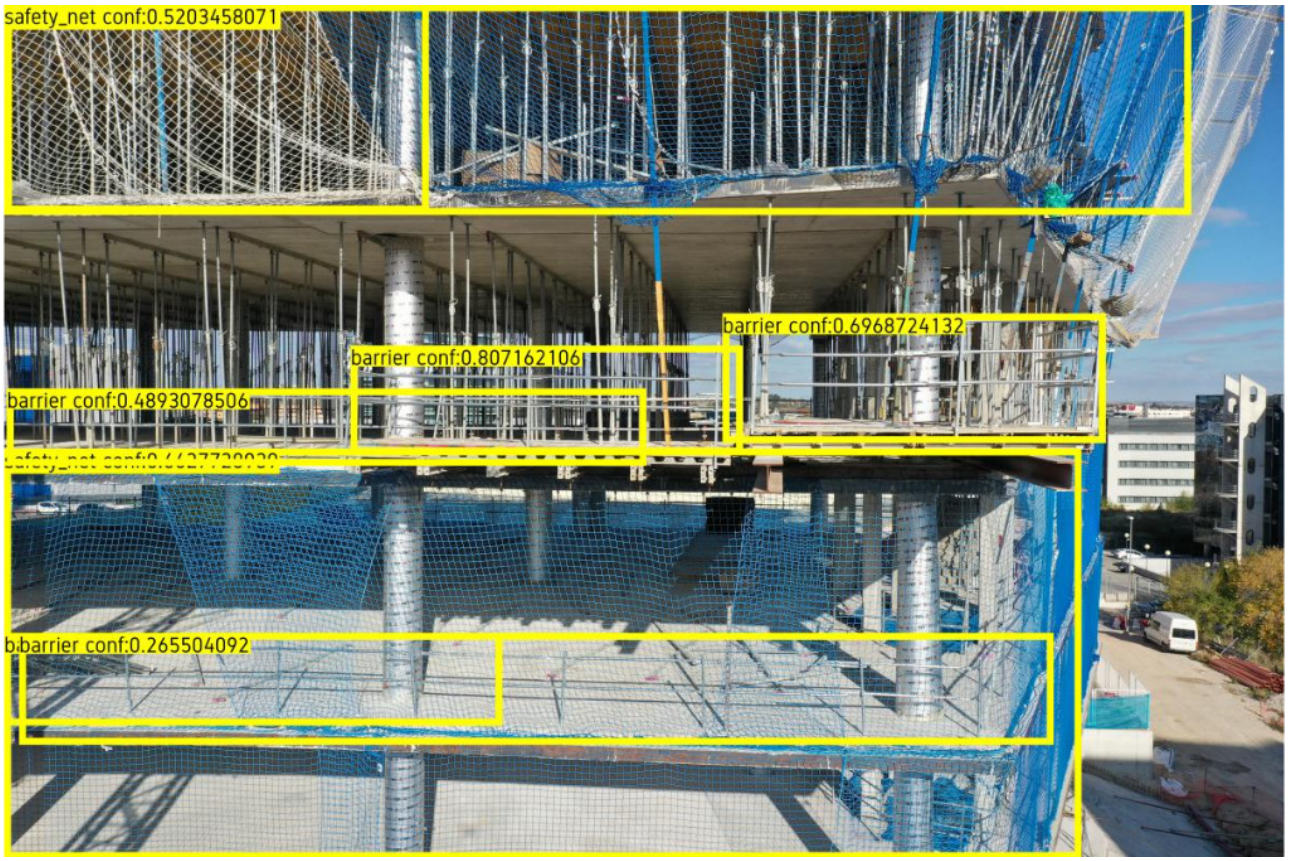

Figure 11 image recognition endpoint example

In the Figure 11 the annotations for each object is shown. The object type and the number indicates certainty (0 to 1).

7. References

[1] GitHub - mavlink/mavlink: Marshalling / communication library for drones, https://github.com/mavlink/mavlink

[2] Developing depth sensing applications - collision avoidance, object detection, volumetric capture and more, https://www.intelrealsense.com/sdk-2/

[3] Camera Remote SDK | SONY, https://support.d-imaging.sony.co.jp/app/sdk/en/index.html

[4] ROS/Technical Overview - ROS Wiki, http://wiki.ros.org/ROS/Technical%20Overview

[5] ROS/TCPROS - ROS Wiki, http://wiki.ros.org/ROS/TCPROS

[6] ROS/UDPROS - ROS Wiki, http://wiki.ros.org/ROS/UDPROS