

---

# **GEOINFORMATICS PROJECT**

## **Plugin Report**

**Version 1.0**

**Project name: Drone Surveying Planning  
(DSP)**



**Prepared by: Marta Rossi, Omar Abdelgafar**

## Table of Contents

1 Introduction.....	3
1.1 Purpose .....	3
1.2 DSP plugin introduction .....	3
2 Python Plugins in QGIS .....	3
2.1 Plugin builder .....	4
2.2 Plugin Reloader .....	4
2.3 Essential files and directory structure of a QGIS plugin .....	4
2.4 QtQGIS .....	5
2.5 Qt.....	5
2.5.1 PyQt.....	5
2.5.2 Qt Designer.....	5
3 DSP plugin installation.....	6
4 GUI description.....	7
4.1 Main dialog window .....	7
4.1.1 Drone Type .....	8
4.1.2 Sensor .....	9
4.1.3 Surveying Area .....	10
4.1.5 Planning Parameters .....	12
4.1.6 Models for Accuracy Prediction .....	12
4.1.6 RUN, CLOSE, HELP buttons.....	13
4.2 Outputs window.....	13
4.2.1 Overlapping Map and Error Map .....	15
4.2.2 Export .....	16
5 Errors .....	16

## 1 Introduction

Before every in-field surveying, a proper setting of all the parameters which are involved is of fundamental importance. The main aim of the Drone Surveying Plugin (DSP) plugin, which combines its functionalities with those of QGIS consists in helping the user to choose the best parameters for the surveying planning, as a prototype to see whether he could plan a drone flight in an efficient way through a usable software.

### 1.1 Purpose

This document is intended to give a detailed explanation of the DSP plugin as well as giving a good overview of the principal functionalities, which are the following:

- setting of the optimal parameters in a user-friendly GUI
- facilitating the definition and visualization of user data using the support of QGIS
- obtaining graphic visualization of overlapping maps and error maps
- save and share results.

### 1.2 DSP plugin introduction

Drone Surveying Planning (DSP) is a QGIS plugin reading some user's input parameters and performing three flight planning algorithms at different level of approximation, which are: Normal case, Simulation of ground control points using the least square adjustment and Simulation using DTM. It allows the user to design the aerial flight as well as getting as output a map of error's standard deviations.

It is a student project related to PHOTOGRAMMETRY AND DRONE SURVEYING course in Politecnico di Milano.

## 2 Python Plugins in QGIS

QGIS functionalities can be extended using plugins, which can be written in Python. For this aim, Python version 3 has been used.

The essential steps to create a plugin in QGIS have been the following:

1. *Idea*: Topic of the plugin
2. *Create files*: Create some files, some are essentials, and define the directory structure (see section 2.3)
3. *Write code*: Write the python code in the appropriate files.


4. Test the plugin by reloading it to check if everything is working; this can be accomplished using another QGIS plugin, called “Plugin Reloader” (see section 2.2)
5. *Publish*: Publish the plugin in QGIS repository or make an own repository.

First, two QGIS plugins enabled the creation and reload of the plugin, which are called “Plugin Builder” and “Plugin Reloader” (see sections 2.1, 2.2).

The Python API called “PyQGIS” links QGIS to Python (described in section 2.4) and “Qt Designer” is used as widget toolkit for creating the graphical user interface (see section 2.5.1).

## 2.1 Plugin builder

The plugin template has been created using a QGIS plugin called “Plugin Builder 3”, which is the recommended option, as it produces 3.x compatible sources. To install or activate this

plugin, go to **Plugins** menu and select  **Manage and install plugins...** or see the following link: <https://plugins.qgis.org/plugins/pluginbuilder3/>.

## 2.2 Plugin Reloader

During development of the plugin, it is frequently needed to reload it in QGIS for testing. This has easily been accomplished using the Plugin Reloader plugin. To install or activate this plugin,

go to **Plugins** menu and select  **Manage and install plugins...** .

## 2.3 Essential files and directory structure of a QGIS plugin

The meaning of the essential files is explained in the following:

- **\_\_init\_\_.py** = The starting point of the plugin. It has to have the classFactory() method and may have any other initialisation code.
- **mainPlugin.py** = The main working code of the plugin. Contains all the information about the actions of the plugin and the main code. In the DSP plugin it is made of three main files, which are the following:
  - o DSP.py
  - o DSP\_dialog.py
  - o survey\_planning.py
- **resources.qrc** = The .xml document created by Qt Designer. Contains relative paths to resources of the forms.
- **resources.py** = The translation of the .qrc file described above to Python.
- **form.ui** = The GUI created by Qt Designer. In DSP plugin they are the followings:
  - o DSP\_dialog\_base.ui
  - o HELP\_dialog.ui
  - o DSP\_outputs.ui
  - o NewDrone.ui
  - o NewSensor.ui

- progress\_msg.ui
- **form.py** = The translation of the form.ui described above to Python.
- **metadata.txt** = Contains general info, version, name and some other metadata used by plugins website and plugin infrastructure.

These basic files (skeleton) of a typical QGIS Python plugin can be created from here:

<https://github.com/wonder-sk/qgis-minimal-plugin>.

Moreover, for this plugin some additional files have been added, which are the following:

- drone\_list.json
- sensor\_list.json

They are JSON files that contain data of predefined types of drones and sensors.

## 2.4 PtQGIS

PyQGIS refers to the python scripting in QGIS. It makes possible to create Python scripts for automating geographic processes and accessing or manipulating QGIS interface; this can be accomplished both using the python console in QGIS or by importing the qgis.core module in the python script. In general, it is a Python API provided by QGIS which also integrates with PyQt.

## 2.5 Qt

Qt is a free and open-source widget toolkit for creating graphical user interfaces as well as cross-platform applications.

QGIS is built using the Qt platform.

Both QT and QGIS itself have well-documented APIs that should be used when writing Python code to be run within QGIS

### 2.5.1 PyQt

PyQt is a set of Python bindings for Qt application framework and runs on all platforms supported by Qt including Windows, macOS, Linux, iOS and Android. The bindings are implemented as a set of Python modules and contain classes which are used to connect the Qt C++ cross-platform framework with the Python language.

### 2.5.2 Qt Designer

Qt designer is a Qt tool for designing and building GUIs with Qt Widgets for PyQt applications. It makes easy to compose and customize dialog windows using different styles. (see Figure 1)

Widgets and forms created with Qt Designer integrate with the programmed code, using Qt's signals and slots mechanism, so that it can be easily assigned a proper behaviour to the graphical elements. All the properties set in Qt Designer can be changed dynamically within the code. This is how the dialog windows have been designed, which are the following:

- **DSP\_dialog\_base.ui**: it is the main dialog window which opens once the plugin is started.

- **HELP\_dialog.ui**: this is a window that the user can open by clicking on the “HELP” button in the main window to retrieve useful information about the plugin main buttons and pop-up errors.
- **DSP\_outputs.ui**: this is the window containing the computed results, which automatically appears after the “RUN” button in the main dialog is clicked to compute a chosen algorithm with specific input parameters.
- **NewDrone.ui**: this is the dialog concerning the setting of the drone parameters used in the surveying, which opens clicking on the “New Drone” button in the main dialog window.
- **NewSensor.ui**: this is the dialog for the sensor’s parameters setting, once the “New Sensor” button is clicked in the main dialog window.
- **progress\_msg.ui**: this is a small window which opens when the computations are running and automatically closes once the computation finishes and the outputs dialog opens.

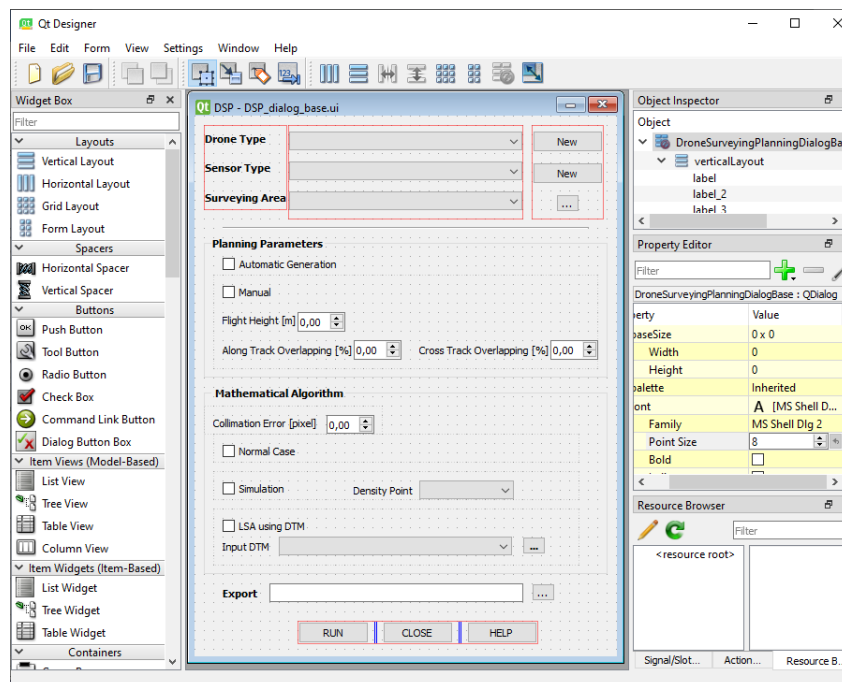


Figure 1

### 3 DSP plugin installation

Since the plugin has not already been published in QGIS, the following procedure must be followed to retrieve the DSP plugin in the user’s QGIS.

Download the plugin scripts from the GitHub repository: <https://github.com/Marta1Rossi/DSP>.

Among them, two .bat files are present: put the resources in the QGIS folder (OSGeo4W64), open them as text files and update a proper path.

Open pyqgis.bat with double click and command line opens.

Then, make sure it's updated pip (=is a python install package):

```
python3 -m pip install --upgrade pip.
```

Then install in QGIS the Plugin Builder tool (which is a compiler) mentioned in Section 2.1.

Next, write:

```
python3 -m pip install pb_tool
```


This is the python interface for Plugin Builder.

Open pyqgis.bat file with double click and the terminal will open and go to your DSP files directory:

```
cd path-to-the-DSP-folder
```

Finally, write in the command line:

- pb\_tool deploy
- y

Then close and reopen QGIS application and go to **Plugins** menu and select  **Manage and install plugins...** .

"DSP" plugin will appear.

## 4 GUI description

### 4.1 Main dialog window

Once the user opens the DSP plugin in QGIS, the main dialog of the GUI will be shown (Figure 2):

The image shows a software window titled 'DSP' with a close button (X) in the top right corner. The window is divided into several sections:

- Drone Type:** A dropdown menu showing 'Pioneer' and a 'New' button.
- Sensor Type:** A dropdown menu showing 'Sens1' and a 'New' button.
- Surveying Area:**
  - Load Imagery:** Three radio buttons for 'Google Satellite', 'ESRI Satellite', and 'Bing'.
  - Shapefile:** A dropdown menu and a file selection button (...).
  - Surveying Area Dimension:** Two input fields for 'X [m]' and 'Y [m]', both set to '0'.
- Planning Parameters:**
  - Automatic Generation:** A radio button. Next to it are input fields for 'GSD [m]: 0,0000' and 'Target az [mm]: 0'.
  - Manual:** A radio button. Below it are input fields for 'Flight Height [m]: 0', 'Along Track Overlapping [%]: 0', and 'Cross Track Overlapping [%]: 0'.
- Models for accuracy prediction:**
  - Collimation Error [pixel]:** An input field set to '1'.
  - Normal Case:** A radio button. To its right is a 'Density Point' dropdown menu set to 'Medium'.
  - Simulation (without DTM):** A radio button. To its right is a 'Density Point' dropdown menu set to 'Medium'.
  - Simulation using DTM:** A radio button. To its right is an 'Input DTM' dropdown menu and a file selection button (...).

At the bottom of the window are three buttons: 'RUN', 'CLOSE', and 'HELP'.

Figure 2

In it, different sections can be distinguished, which are listed below.

#### 4.1.1 Drone Type

The upper part of the GUI represents a section dedicated to the collection of parameters concerning the drone, sensor, and surveyed area.

As for the drone, the user can choose to select a default drone type or to insert a new one.

- Default drone: the names of 2 default drone models are listed in the combo box. By selecting one of them, its parameters are directly considered as input parameters by the application. Their values are retrieved from the "drone\_list.json" JSON file. The user can check the values of the default drone models by opening the HELP window, by clicking the "HELP" button at the bottom of the GUI.
- For inserting a new drone type, the user must click on the "New" button: a dedicated window called "Drone" will open, as shown in Figure 3.



Figure 3

In this window, the drone parameters can be set, which are the following:

- the Drone Name
- the drone Maximum Altitude in meters
- the drone/UAS (Unmanned Aerial Vehicle) Maximum speed in km/h
- the drone's Battery Duration in minutes.

Once the user clicks on “Ok” button, this window will automatically close and the chosen drone name will appear in first position in the list of the drone type combo box, to be considered as well as its parameter values as inputs for the application.

#### 4.1.2 Sensor

The same configuration is valid for the sensor:

- the user can choose a default sensor type, among the combo box list, whose parameter values can be checked in the HELP window.
- the user can create a new sensor type to be used for the survey, clicking on “New” button which opens a new dialog called “Sensor” (figure x) and setting the values of its parameters, which are listed below:

- Sensor Name
- Focal Length in mm
- Shooting Interval in seconds
- Sensor Size in X (width) in mm
- Sensor Size in Y (height) in mm
- Image Size in X (width) in pixels
- Image Size in Y (height) in pixels

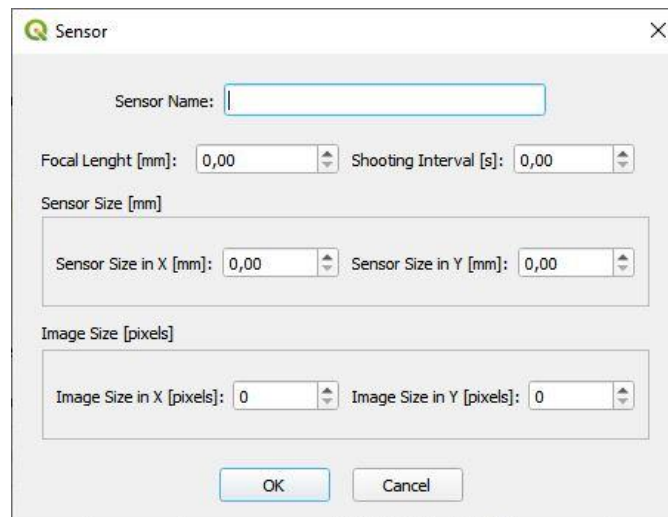


Figure 4

By clicking on “Ok” button, the sensor window automatically close and the set sensor’s name is displayed in the sensor’s combo box in the main window.

Again, the specific values of the default sensor types can be found in the “HELP” window.

#### 4.1.3 Surveying Area

For the Surveying Area the user can both automatically retrieve the area dimensions in three ways:

- 1) importing a shapefile to automatically retrieve the area dimensions (X, Y)
  - 2) creating a new shapefile drawing it on the available web basemaps
  - 3) setting the area dimensions manually.
- 
- 1) If retrieving the area dimension automatically, the user must provide a shapefile to the application. This can be done both loading a vector layer in QGIS table of contents (TOC) before opening the DSP plugin, such that its name will automatically appear in the Surveying Area combo box (Figure 5) or loading it directly from the plugin by using the browse. After a valid shapefile is loaded, the application will automatically retrieve the area dimensions in meters (X, Y) and the values will be displayed.
  - 2) “Load Imagery” let the user choose among three web basemaps which are Raster Tile layers that can be displayed on QGIS through a radio button, which are the following:
    - Google Satellite
    - ESRI Satellite
    - Bing Satellite

In this way the user can draw a new shapefile in the Region Of Interest (ROI) and its name and area dimensions will automatically appear in the GUI. (See Figure 6)
  - 3) Otherwise, the surveying area dimensions can be written manually inside the double spin boxes.

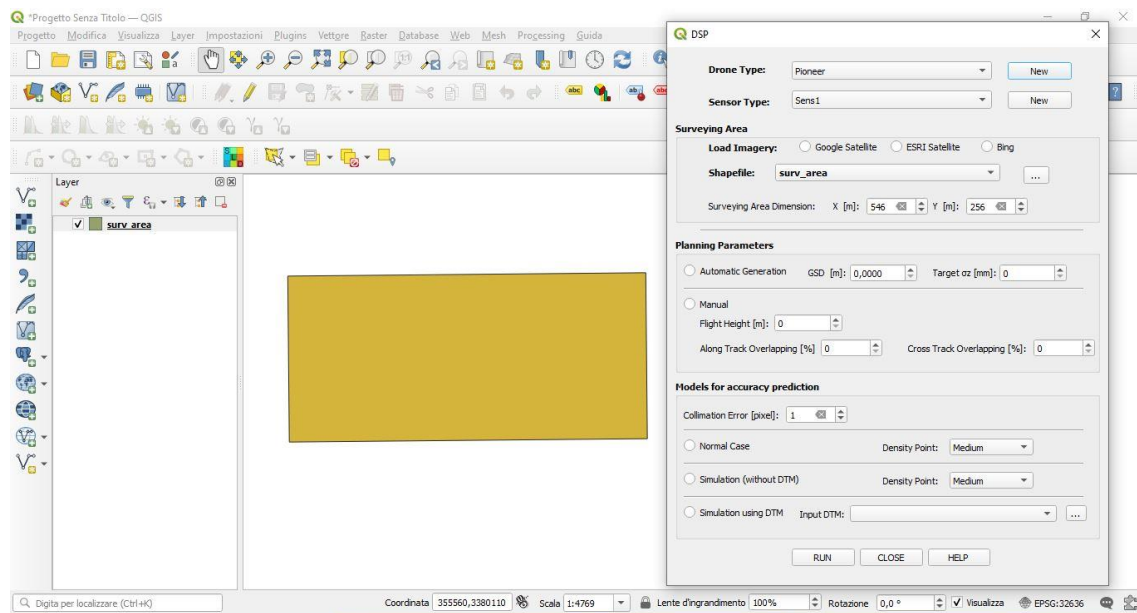


Figure 5

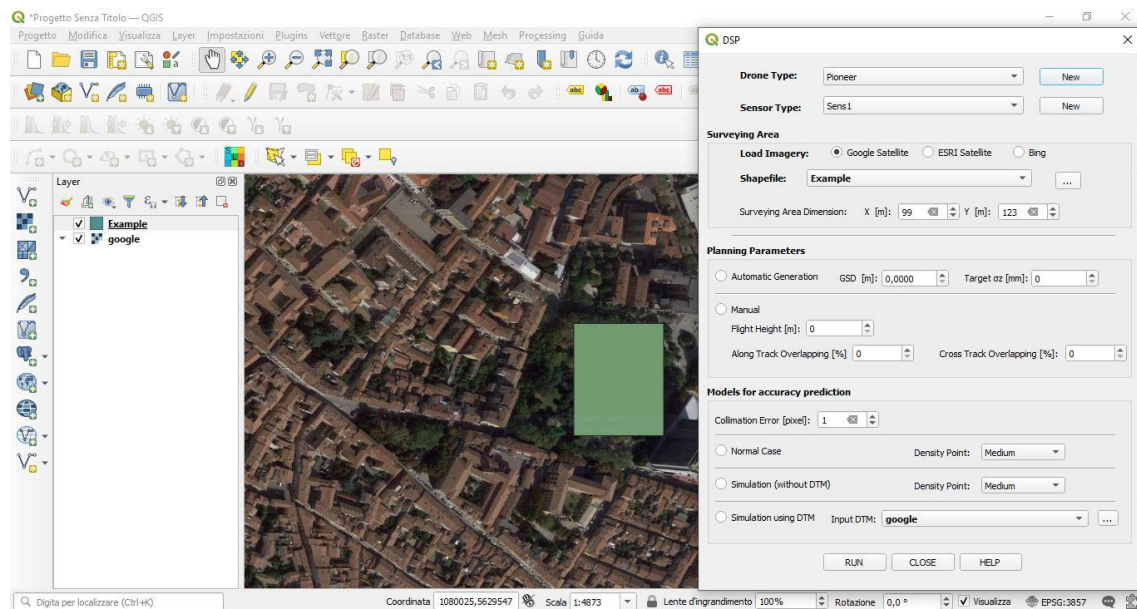


Figure 6

Note that the shapefile can have both rectangular and polygonal shape, the code will automatically compute the width (X) and height (Y) dimensions by considering its vertices.

#### 4.1.5 Planning Parameters

In the “Planning Parameters” section, the user is asked to choose between two options, by clicking on the corresponding radio button:

- 1) “Automatic Generation”: the user is going to provide to the application the value of the Ground Sampling Distance (GSD) in meters and the value of the error’s variability in height direction ( $\sigma_z$ ) in mm representing the target value for the specific surveying; by considering these values as constraints, the application will find the optimal planning parameters for the user, which are the following:
  - Flight Height in meters
  - Along Track and Cross Track Overlapping percentage.These values will be shown to the user together with all the results in the output window (see Figure 6), once all the other requested input variables are set and the “Run” button is clicked.
- 2) “Manual”: the user provides the values of the planning parameters (Flight Height, Along Track and Cross Track Overlapping).

#### 4.1.6 Models for Accuracy Prediction

The last section of the main dialog, called “Models for accuracy prediction”, concerns the choice of the prediction algorithm to be used to perform the survey. First, the value for the Collimation Error in pixel units is required, regardless the choice of the algorithm; the default value is already displayed and corresponds to 1. The user can select the model for the accuracy prediction by clicking on the corresponding radio button.

Note that for both the “Normal Case” and “Simulation (without DTM)” methods, a choice for the density of the simulated ground points is provided through a combo box; the possible values are the following: “Low”, “Medium” (default one), “High”. The user can come to know more information about the meaning of these values by clicking on the “HELP” button. The point density value is found by dividing the shortest area dimension by the factors: 15, 25, 35 for the Low, Medium, and High labels, respectively.

Finally, for the “Simulation using DTM” model, both a shapefile and a DTM are required. The shapefile is set in the same way as previously mentioned (see section 3.1.3), while the DTM can be automatically collected by the plugin in case it is already present in the QGIS’ TOC or it can be imported in the GUI using the browse button.

Note that in case a polygonal shapefile is given to the GUI, the plugin is going to obtain the corresponding oriented rectangular area, starting from its vertices and will save this new vector layer in the same directory of the original polygonal one adding a specific name (“rectangle\_area”). After, the rectangular shapefile will be used by the plugin to cut the same portion of the underlying DTM, whose values will then be used as input for the selected “Simulation method using DTM”. The cutted DEM will be saved in tiff format in the same folder of the original raster layer, with name “cutted\_DEM”.

#### 4.1.6 RUN, CLOSE, HELP buttons

In the lower part of the GUI the three following buttons are displayed:

- RUN: the user will click on it once all the input parameters are set. If some of them are missing or are not accepted by the plugin, some popups will appear providing the description of the occurred error. When all the input parameters are set in the correct way, by clicking this button the computation of the algorithms is triggered and the window of the outputs appears.
- CLOSE: closes the plugin GUI.
- HELP: the user can open a window where some relevant explanation about the GUI sections and input parameters is given, as well as an explanation of the source of the popup errors. (See Figure 7)

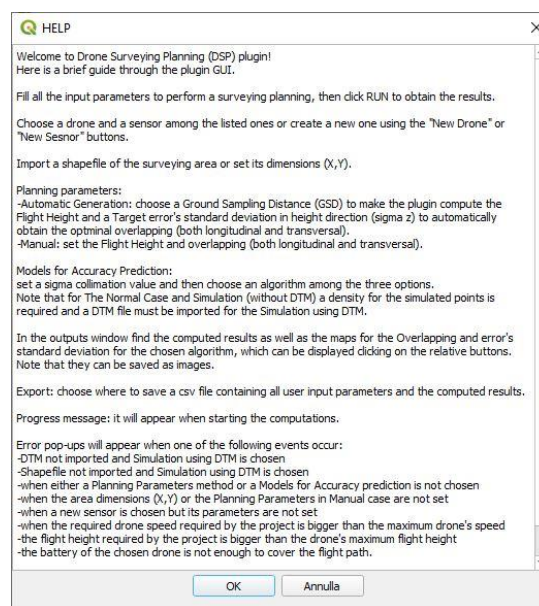


Figure 7

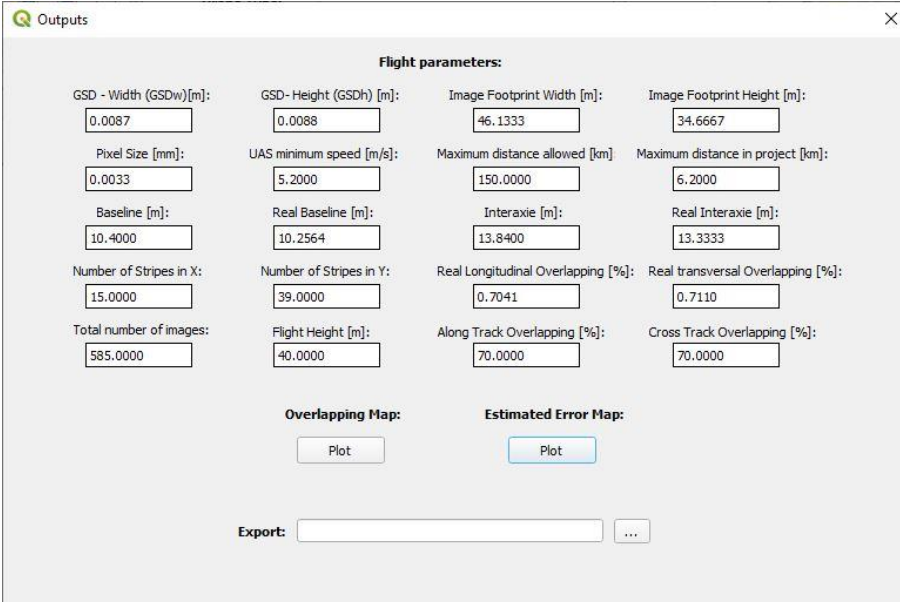
#### 4.2 Outputs window

After the user sets all the input parameters and clicks on "RUN" button, the plugin starts calculating the results which will appear in the outputs dialog once the computation is finished. (Figure 8)

The computed parameters are the following, and they are the same for any of the three algorithms:

- Ground Sample Distance (GSD) [m]
- Image Footprint [m]
- Pixel size [mm]
- UAS minimum speed [m/s] → compliant with chosen sensor 'shooting interval'

- Maximum distance allowed [km] → which can be covered by the chosen drone type
- Maximum distance in project [km] → length of the flight path
- Baseline [m] → distance between the camera centres of projection, along track
- Real Baseline [m] → value adapted to uniformly cover the area
- Interaxie [m] → distance between the camera centres of projection, cross track
- Real interaxie [m] → value adapted to uniformly cover the area
- Number of Stripes → followed by the drone in the project, for both directions (X, Y)
- Real Longitudinal and Transversal overlapping [%] → set by user in “Manual method” or computed and suggested by the plugin in “Automatic Generation” case.
- Total number of images → shot by the drone in the surveying project
- flight height (h [m]), computed in the automatic generation method or the same as the user input in the manual planning
- Longitudinal and transversal overlapping → set by user in “Manual method” or computed and suggested by the plugin in “Automatic Generation” case.



The screenshot shows a software window titled "Outputs" with a close button (X) in the top right corner. The main content area is titled "Flight parameters:" and contains a grid of input fields with numerical values. Below this grid are two sections: "Overlapping Map:" and "Estimated Error Map:", each with a "Plot" button. At the bottom, there is an "Export:" label followed by a text input field and a three-dot menu button.

Flight parameters:			
GSD - Width (GSDw)[m]:	GSD - Height (GSDh) [m]:	Image Footprint Width [m]:	Image Footprint Height [m]:
0.0087	0.0088	46.1333	34.6667
Pixel Size [mm]:	UAS minimum speed [m/s]:	Maximum distance allowed [km]:	Maximum distance in project [km]:
0.0033	5.2000	150.0000	6.2000
Baseline [m]:	Real Baseline [m]:	Interaxie [m]:	Real Interaxie [m]:
10.4000	10.2564	13.8400	13.3333
Number of Stripes in X:	Number of Stripes in Y:	Real Longitudinal Overlapping [%]:	Real transversal Overlapping [%]:
15.0000	39.0000	0.7041	0.7110
Total number of images:	Flight Height [m]:	Along Track Overlapping [%]:	Cross Track Overlapping [%]:
585.0000	40.0000	70.0000	70.0000

Overlapping Map:

Estimated Error Map:

Export:

Figure 8

4.2.1 Overlapping Map and Error Map

These two show the user the plotted overlapping graph (See Figure 9) and errors (see Figure 10) for the chosen algorithm, which can also be saved as image.

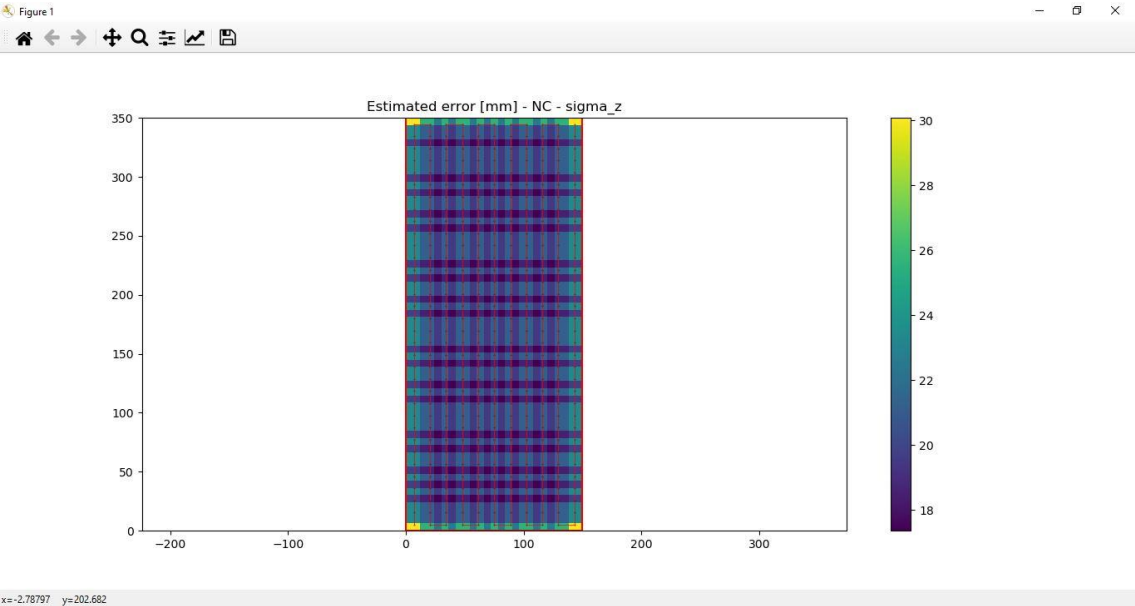


Figure 9

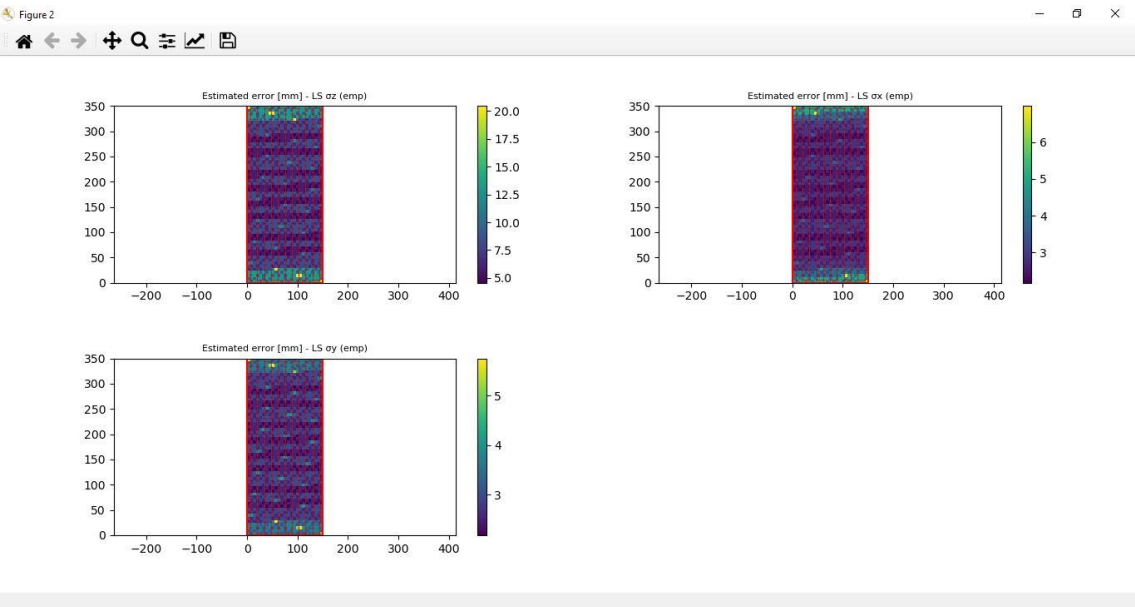


Figure 10



#### 4.2.2 Export

The results, together with the user inputs, can be automatically saved in a csv file.

### 5 Errors

Error pop-ups will appear when one of the following events occurs:

- DTM not imported and Simulation using DTM is chosen
- Shapefile not imported and Simulation using DTM is chosen
- when either a Planning Parameters method or a Models for Accuracy prediction is not chosen
- when the area dimensions (X, Y) or the Planning Parameters in Manual case are not set
- when a new sensor is chosen but its parameters are not set
- when the required drone speed required by the project is bigger than the maximum drone's speed
- the flight height required by the project is bigger than the drone's maximum flight height
- the battery of the chosen drone is not enough to cover the flight path.

Some examples of their design are shown in Figure 11.

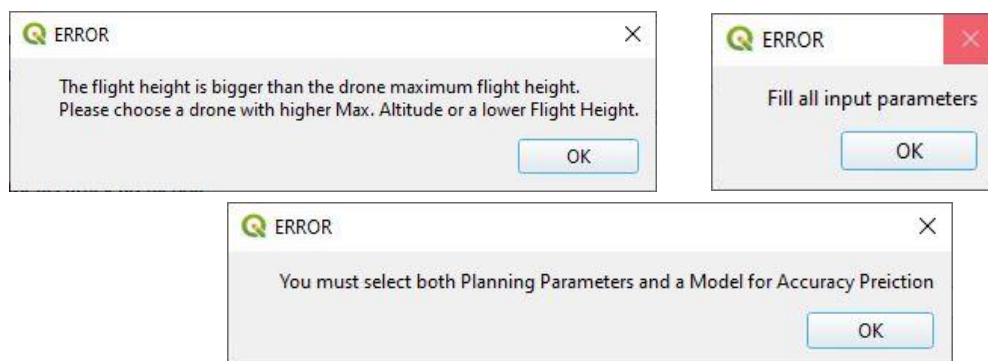


Figure 11