Norwegian Institute for Water Research

8116-2025

Overview of the SeaBee data platform

Report

Norwegian Institute for Water Research STI

Serial no: 8116-2025

ISBN 978-82-577-7853-8
NIVA report
ISSN 1894-7948

This report has been quality assured according to NIVA's quality system and has been approved by:

James Sample
Project Manager

Dag Hjermann
Quality Assurer

Dag Hjermann
Research Manager

www.niva.no

Title
Overview of the SeaBee data platform

Pages
20

Date
23.06.2025

Author(s)
James Sample, Kim Leirvik, Debhasish Bhakta, Andrea Merlina

Topic group
Marine biology

Distribution
Open

Client(s)
NFR

Published by NIVA
Project number 200064

Abstract
SeaBee is the Norwegian national infrastructure for drone-based research, mapping and monitoring, funded by the Research Council of Norway. This report provides a high-level overview of the software, tools and data pipelines developed during Phase 1 of the SeaBee project (2020 to 2025).

Keywords: SeaBee, Drones, Data processing, Remote sensing
Emneord: SeaBee, Droner, Databehandling, Fjernmåling

Table of contents

Abbreviations

AI	Artificial intelligence
API	Application programming interface
CI	Continuous integration
CNN	Convolutional Neural Network
CPU	Central processing unit
DSM	Digital surface model
DTM	Digital terrain model
FAIR	Findable, accessible, interoperable, reusable
GB	Gigabyte
GCP	Ground control point
GIS	Geographic information systems
GPS	Global positioning system
GPU	General processing unit
HSI	Hyperspectral imagery
IDE	Interactive development environment
IMU	Inertial measurement unit
LiDAR	Light Detection and Ranging
ML	Machine learning
MSI	Multispectral imagery
NFR	Research council of Norway (Norges forskningsråd)
NINA	Norwegian institute for nature research
NIRD	National infrastructure for research data
NIVA	Norwegian institute for water research
NR	Norwegian computing centre (Norsk regnesentral)
NTNU	Norwegian university of science and technology
ODM	Open drone map
PUE	Power usage effectiveness
RGB	Red, green, blue
RTK	Real time kinematic
TB	Terabyte
UI	User interface
VM	Virtual machine
WFS	Web feature service
WMS	Web mapping service
WUE	Water usage effectiveness

1 Introduction

SeaBee is the Norwegian national infrastructure for drone-based research, mapping and monitoring. The project is funded by the Research Council of Norway (NFR; project number 296478) and aims to establish a centre for drone-based coastal research, including mapping of habitats, monitoring animal communities, and assessing anthropogenic impacts.

The SeaBee project has two phases: Phase I (2020 – 2025) is funded directly by NFR to establish the infrastructure, including purchasing hardware, training drone pilots, and developing protocols and software for data processing; Phase II (2025 – 2030) is an operational phase where SeaBee services will be made available to other projects and user groups.

This document provides an overview of the software, tools and data pipelines established during Phase I of the project. These are collectively referred to as the "SeaBee data platform", which aims to provide researchers with an end-to-end solution for processing, analysing and sharing data collected by drones.

2 Scope and aims

The SeaBee data platform is primarily designed to process aerial imagery collected by flying drones. SeaBee drones are equipped with a variety of sensors, the most common being RGB (red, green, blue) and multispectral (MSI) cameras. Less commonly, dedicated thermal (infrared) cameras may be used for population surveys (e.g. for seabirds or seals). Raw data from these sensor types typically comprises large numbers (hundreds to thousands) of geotagged, overlapping images, which must be "stitched" together to create an orthophoto mosaic prior to further processing.

Other Seabee drones are equipped with hyperspectral (HSI) cameras, which have lower resolution than RGB or MSI sensors but gather more detailed spectral information. These sensors use line-scanning (sometimes called "push-broom" technique) to collect data, and the raw output is a series of strips or swathes, rather than individual camera frames. For this reason, the data processing pipeline for HSI data is separate from the pipeline for RGB and MSI data.

A core aim of the SeaBee platform is to allow drone pilots to upload raw data directly from the field and have it processed, analysed and published online within a matter of hours. Although SeaBee does not explicitly aim to provide results in "real-time", fast and efficient processing is valuable because it allows pilots to review results in the field and adjust flight settings accordingly.

In addition to automated pipelines, the SeaBee platform aims to make SeaBee data FAIR (Findable, Accessible, Interoperable and Reusable) by providing researchers with tools for visualising, sharing and analysing SeaBee data products. Users can include SeaBee data in their own workflows or build third-party applications that consume SeaBee services. The SeaBee platform provides a variety of tools targeted towards different user groups:

- **External users** can search and browse SeaBee datasets online, share links to specific datasets, and use SeaBee services in their own applications (such as desktop GIS).
- **Registered users** have additional options to download raw data and access web GIS interfaces where they can create customised, sharable maps comprising multiple data layers. Registration

is free and without obligation. Registered users can also request access to the SeaBee data portal, where they can upload their own drone data for automatic processing.

- **Advanced users** have access to sophisticated data analysis tools and computational resources allocated on the SeaBee platform itself. These users can write code to analyse and visualise SeaBee data, without having to install or download anything locally. Advanced users can share their code on the platform, create new platform components, and develop entirely new processing pipelines.

A secondary goal of the SeaBee platform is to host ancillary datasets associated with drone flying operations. These include "ground truth" datasets collected by field staff for training machine learning algorithms, as well as data from other kinds of drone (boats, submersibles, etc.) and sensor (e.g. LiDAR). The range of datasets supported by the platform is expected to expand over time in response to user and project needs.

Drone hardware, artificial intelligence (AI) and machine learning (ML) are all developing rapidly. A core goal of the SeaBee project is therefore to provide a flexible and extensible data platform that can adapt to accommodate new data sources (drones and sensors) and algorithms as they become available.

3 Hosting and data centre

The SeaBee data platform is hosted by Sigma2 as part of Norway's National Infrastructure for Research Data (NIRD). NIRD provides data storage and cloud computing infrastructure specifically tailored to the scientific community. The SeaBee platform runs on NIRD's Data Peak service, with daily backups to the Data Lake.

All SeaBee datasets are stored in Sigma2's data centre at Lefdal Mine in Nordfjord. Formerly one of the largest olivine mines in the world, it was converted to a data centre in 2018 and SeaBee data migrated at the start of 2023. The centre comprises 6 levels with a total usable space of 120 000 m^2 (*Figure 1*).

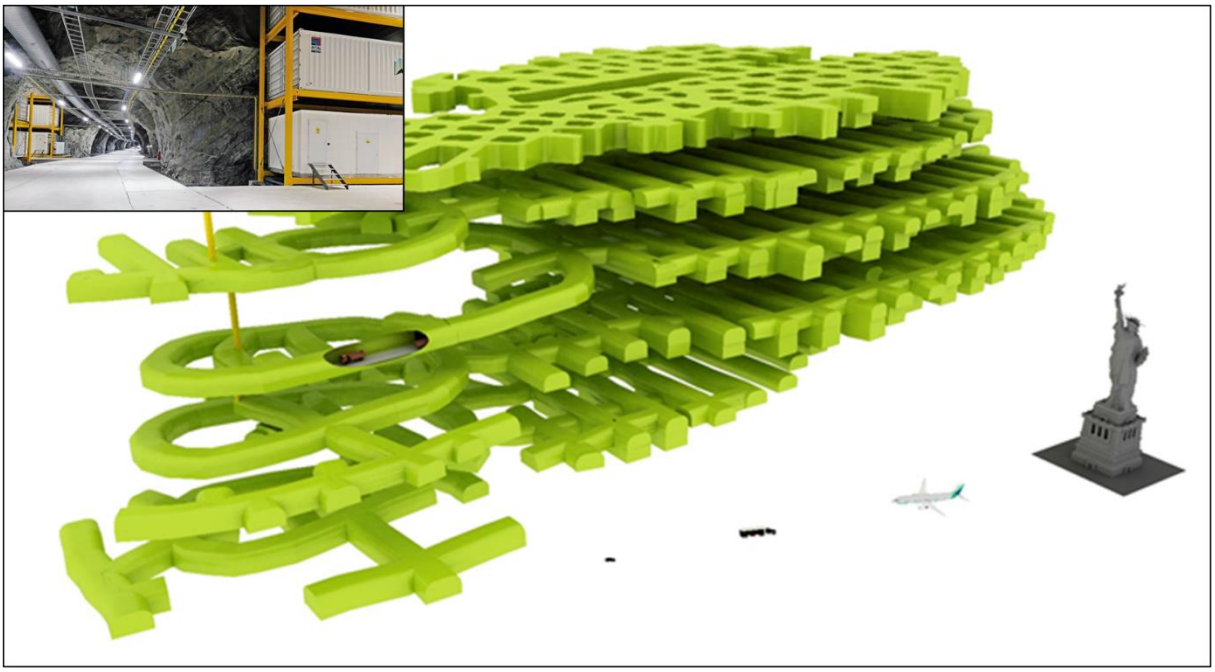

Figure 1: Lefdal Mine Data Centre in Nordfjord. Images © Sigma2.

Lefdal Mine is one of the greenest data centres in Europe, with a strong commitment to environmental stewardship and sustainability. The facility operates on renewable energy from a nearby hydroelectric plant and takes cooling water from deep in the neighbouring fjord. This ensures a minimal carbon footprint and eliminates the need for energy-intensive evaporative cooling. As a result, the centre achieves a Water Usage Effectiveness (WUE) of zero and a Power Usage Effectiveness (PUE) of 1.15, making it one of the most efficient data centres in the world. Furthermore, by reusing infrastructure from a disused mine, construction costs have been kept low and the visual impact minimal.

A core aim of SeaBee is to promote better environmental management. Lefdal Mine's focus on green energy and sustainable solutions therefore makes it an ideal location to host the SeaBee data platform.

4 Computing environment

4.1 Cloud architecture and resources

The SeaBee data platform operates within a dedicated Kubernetes namespace on NIRD. Kubernetes is an open-source system for automating the deployment, scaling, and management of containerised applications. It offers an industry-standard approach for building cloud-based infrastructure in a provider-agnostic manner.

Sigma2 allocates resources for the SeaBee platform to the Kubernetes namespace. These resources are adjusted to match platform workloads. As of March 2025, the SeaBee namespace has access to 125 TB of storage (plus backup), 1 TB of memory, 110 CPUs (AMD EPYC 7713), and 3 GPUs (Tesla V100). Within the namespace, Kubernetes sets resource limits for different platform components.

Platform components are containerised using Docker. Docker allows developers to package applications and all their dependencies into standardized "units" for software development, ensuring consistency across multiple environments. Docker strikes a balance between deploying applications on a single host and using virtual machines (VMs). Unlike VMs, which include a full operating system and can be resource-intensive, Docker containers share the host system's kernel and run as isolated processes. This makes them more lightweight and efficient while still offering the benefits of isolation and consistency across different environments.

Docker distinguishes between **images** and **containers**. Images are read-only templates that contain the instructions for creating a container. They include everything needed to run an application, such as the code, runtime, libraries, and dependencies. Containers are runnable instances of images. They are created from images and can be started, stopped, moved, and deleted. Each container is isolated from others, providing a secure environment for running applications.

Each component of the SeaBee data platform has an associated **deployment**, handled using Helm or Kustomize. These deployments define containers orchestrated by Kubernetes. Kubernetes monitors the containers, balancing workloads and checking resource consumption. Containers with issues are automatically restarted, while those consuming excessive resources are shut down to prevent impact on other platform components.

Together, Docker and Kubernetes provide a robust and widely used approach for building cloud-based computational infrastructure.

4.2 Code repositories and documentation

SeaBee code is hosted online as part of the SeaBee GitHub Organisation. As far as possible, SeaBee code is open-source and made available to the public under a GNU General Public License v3.0 (GNU GPLv3). Technical documentation for the platform is also available online, including descriptions of platform components, common workflows, and SeaBee data standards.

Users with specific technical problems (bugs etc.) are encouraged to create GitHub Issues in the appropriate SeaBee repository (e.g. here). More general questions or discussions can use GitHub Discussions instead.

4.3 Development and deployment cycle

Core components of the SeaBee data platform are deployed using Continuous Integration (CI) via GitHub Actions and Workflows. When code changes are committed to the '`main`' branch of a SeaBee repository, a GitHub Workflow is typically triggered to rebuild the Docker image associated with that component (see Section 4.1). The new image is then automatically deployed by Kubernetes as a container, replacing older instances with the updated version.

CI streamlines the process of updating and maintaining complex data pipelines. For instance, automatic pipelines on the SeaBee platform generally run every hour. These pipelines can be updated seamlessly without interrupting the schedule by simply pushing code changes to GitHub and waiting for the CI workflows to complete. If the CI workflow is running when the automatic pipeline starts, the platform will continue to run the previous version of the code from the old Docker images. Once the CI workflow finishes and a new image becomes available, the updated code will run automatically next time the pipeline starts.

5 Platform components

Core components of the SeaBee data platform are illustrated in *Figure 2*. The large, dashed outline labelled "Sigma2" identifies the main components deployed within SeaBee's Kubernetes namespace on Sigma2's NIRD platform.

The SeaBee data platform can be considered as five distinct workflows, with a sixth component – the SeaBee JupyterHub – linking them together:

- The **storage workflow** provides tools to access SeaBee datasets stored on NIRD's Data Peak infrastructure.
- The **orthorectification workflow** comprises tools for stitching and mosaicking raw data from flying drones to create orthophotos, Digital Surface Models (DSMs) and other primary data products.
- The **machine learning workflow** applies pre-trained ML algorithms to classify orthophotos, creating secondary data products such as habitat maps, population counts, etc.
- The **annotation workflow** helps ecologists label objects or features of interest in orthophotos. Labelled datasets form the basis for training new ML algorithms, which are then deployed in the ML workflow.

- The **visualisation workflow** comprises the components needed to make SeaBee data products available to external users. These include web GIS interfaces, Web Mapping Services (WMSs), and tools for searching and exploring datasets and metadata.
- The SeaBee **JupyterHub** provides an Interactive Development Environment (IDE) connecting workflows together. Advanced users and system administrators can login to the JupyterHub and develop pipelines to move data from one part of the platform to the next. Automated pipelines use the same computational environment as the JupyterHub, so components developed and tested on the Hub can be easily deployed on a schedule to execute pre-defined tasks.

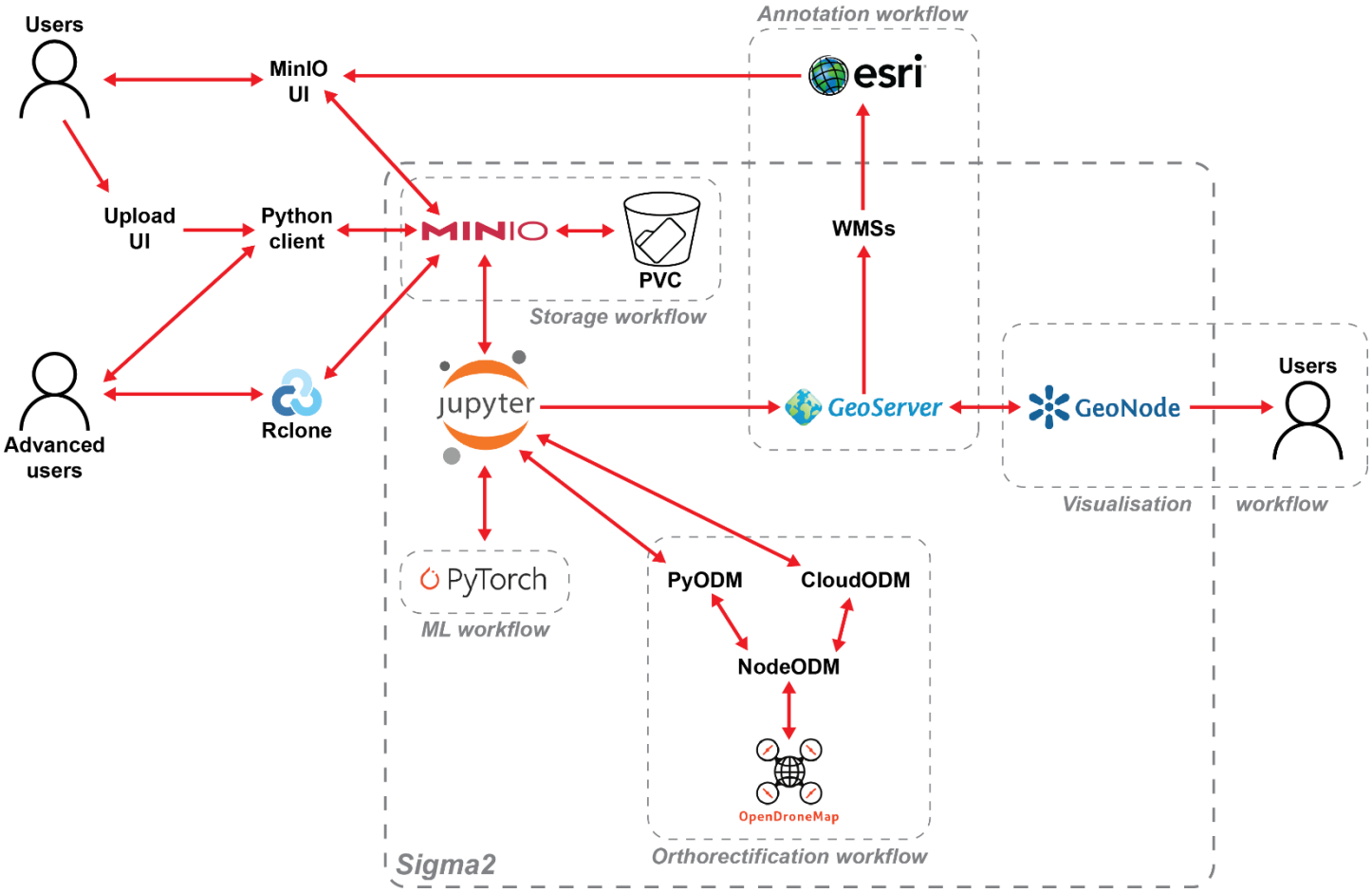

Figure 2: Core components and "workflows" on the SeaBee data platform.

5.1 Storage

The SeaBee platform offers a unified storage solution for all data, complete with backup, and provides different entry points tailored to different user groups.

All SeaBee data is stored on a volume hosted by NIRD's Data Peak service. This volume is mounted directly into HOME directories on the JupyterHub as a folder named '`shared-seabee-ns9879k`', allowing users to easily access SeaBee data as part of their native file system (*Figure 3a*).

The NIRD data volume is also exposed externally via MinIO, an open-source object storage solution that provides an Amazon Web Services S3-compatible API and supports all core S3 features. SeaBee's MinIO

includes an online user interface (UI) where users can browse the SeaBee file system and upload/download files (*Figure 3b*). Additionally, it exposes an S3 storage endpoint (https://storage.seabee.sigma2.no) that can be accessed by any S3-compatible client. *Figure 3c* shows the desktop application RcloneBrowser using this endpoint to access SeaBee data from an external machine, while *Figure 3d* demonstrates programmatic data access using the same endpoint from Python (with seabeepy; see Section **Error! Reference source not found.**).


```python
import seabeepy as sb
from seabeepy.config import SETTINGS

# Login to MinIO
client = sb.storage.minio_login(
    user=SETTINGS.MINIO_ACCESS_ID, password=SETTINGS.MINIO_SECRET_KEY
)

# Copy folder
src_fold = r"/home/notebook/shared-seabee-ns9879k/seabirds/2024/agder-arendal_furuholmen_202406071853"
dst_fold = r"/home/notebook/temp"
res = sb.storage.copy_folder(src_fold, dst_fold, client, containing_folder=True)
```

(d)

Figure 3: Four different options to access the same SeaBee data. (a) JupyterHub file browser; (b) MinIO web interface; (c) RcloneBrowser from an external machine; (d) Programmatically via Python using seabeepy.

Different storage entry points are recommended for different use cases. For instance, the MinIO web UI provides a convenient way to browse SeaBee data and upload or download small to medium-sized files. However, since data transfer occurs over a standard HTTP connection, it is not ideal for transferring large data volumes. By contrast, clients such as Rclone and RcloneBrowser are specifically designed for transferring large volumes of data to and from cloud storage. Unlike standard HTTP data transfers, Rclone tracks progress and can restart if the network connection is interrupted. It is therefore the recommended option for users needing to transfer large volumes of data to/from the SeaBee platform – especially when transferring data directly from the field, where network connectivity can be unreliable.

5.2 Orthorectification and georeferencing

Raw data collected by drones requires processing to create fully georeferenced mosaic images for data analysis. SeaBee's RGB, MSI and thermal sensors capture numerous overlapping images, and the process of combining these into mosaics is called orthorectification. On the SeaBee data platform, orthorectification is performed using Open Drone Map (ODM), an open-source software ecosystem for

processing and analyzing aerial drone data. However, ODM does not support georeferencing data from HSI cameras, because SeaBee's HSI sensors collect data through line-scanning rather than single images. Georeferencing of HSI data is therefore managed on the platform by a separate, custom workflow developed by the Norwegian University of Science and Technology (NTNU; see Section 5.6).

mission_name	n_images	progress	status
Fedje_Langedalen_20230531	203	6.75	RUNNING
Fedje_Ljosoyna_20220609	422	4.50	RUNNING
Bergen_Realfagstaket_20230625	137	6.75	RUNNING
oslofjorden_lyseren-zoom_202306271117	50	6.75	RUNNING
Fedje_Buroyna_20230605	726	0.00	QUEUED
Bjornafjorden_Lyngoya_20230518	879	0.00	QUEUED
Fedje_Lyngholmen_20230605	279	0.00	QUEUED
Fedje_Holmengra_20230605	242	0.00	QUEUED
Fedje_Stormark_20230531	219	0.00	QUEUED
Fedje_nordtrafikkstasjon_20230531	174	0.00	QUEUED

Figure 4: NodeODM task queue showing four missions being processed in parallel.

ODM is deployed on the SeaBee platform using NodeODM and pyodm. NodeODM provides an Application Programming Interface (API) that is used by pyodm to pass drone data to and from the underlying image processing engine. In order to run efficiently, ODM is allocated a significant proportion of the computational resources from the SeaBee Kubernetes namespace. NodeODM maintains a task queue of active jobs and is capable of processing data from multiple drone missions simultaneously (*Figure 4*).

Key data products generated by ODM are orthophotos (georeferenced image mosaics) and DSMs (*Figure 5*). Secondary outputs include point clouds, meshes and texture models.

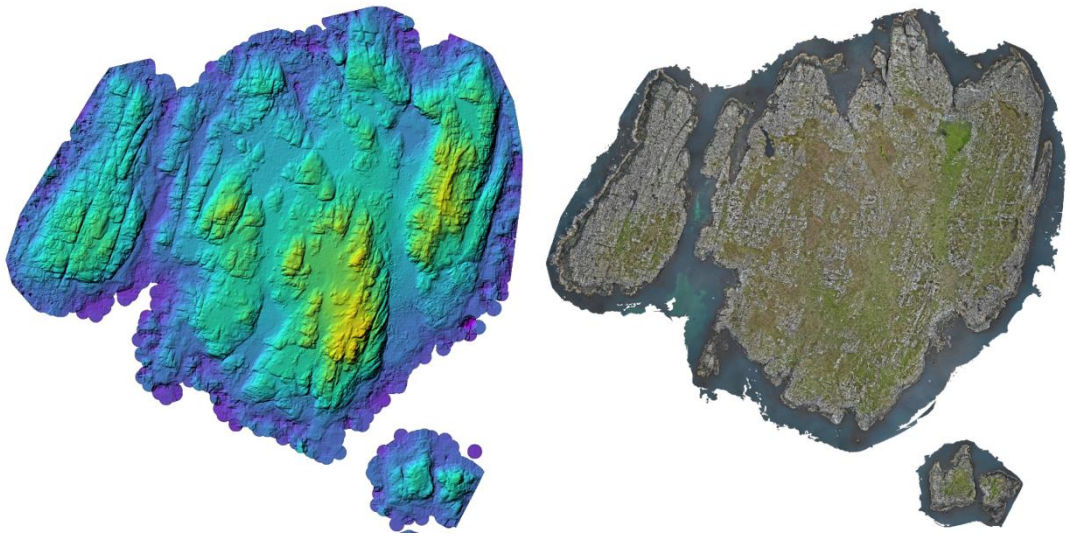

Figure 5: DSM (left) and orthophoto (right) created by ODM from an RGB dataset.

5.3 Image classification using machine learning

The SeaBee data platform uses ML models developed and tested by Norsk Regnesentral (NR). The models are primarily Convolutional Neural Networks (CNNs) implemented in PyTorch and trained using supervised learning with labelled data created by SeaBee ecologists (Section 5.4).

ML models are trained and tested by NR experts using their own hardware. Once trained, the models are deployed "in production" on the SeaBee platform to classify new data. NR delivers models as versioned zip archives and uploads them to SeaBee storage via MinIO. Different models are provided for image segmentation (e.g. habitat classification) and object detection (e.g. counting seabirds). SeaBee's automated pipelines are designed to select the appropriate model based on the classification task.

Outputs from ML models for segmentation (e.g. habitat mapping) are typically raster images showing predicted habitat classes and confidence scores. For object detection (e.g. seabird mapping), the models produce vector datasets with bounding boxes identifying the location of each individual, along with attribute information describing the predicted class (species etc.) and confidence score (*Figure 6*).

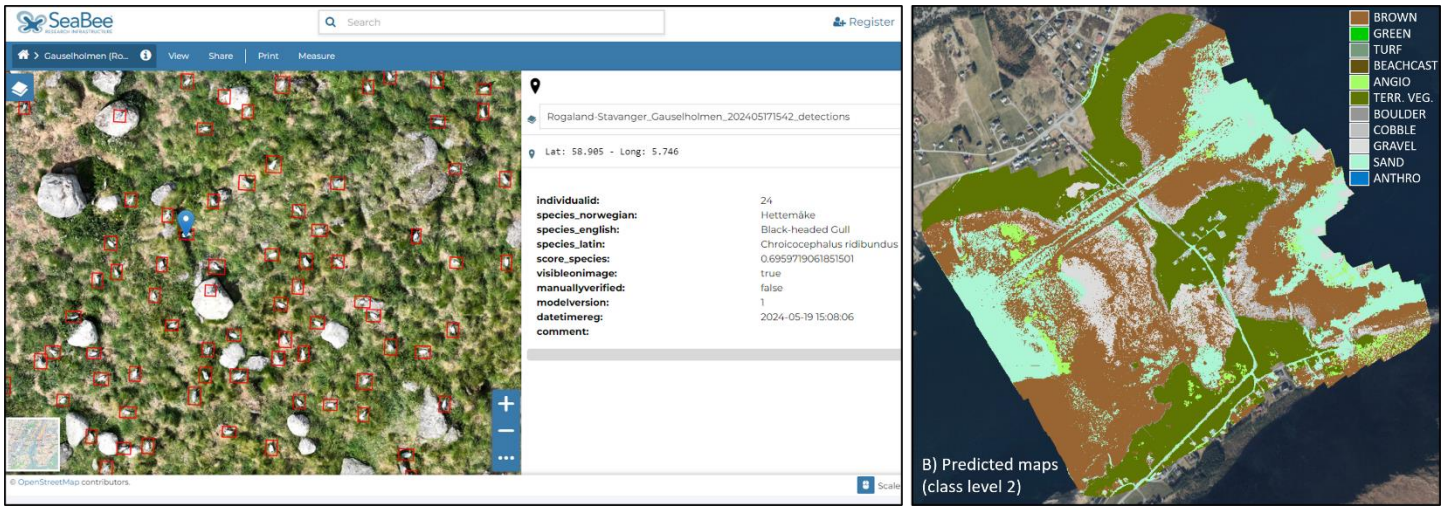

Figure 6: Example ML output for seabird detection (left) and habitat mapping (right).

5.4 Annotation

Training new ML models using supervised learning requires labelled orthophotos identifying features of interest, such as habitat types or bird/mammal species. Orthophotos created by ODM are added to the SeaBee GeoServer (Section 5.5) and made available as WMSs. These can be integrated into desktop GIS or other web applications and annotated by domain specialists.

Annotation of seabird datasets is performed outside the SeaBee platform using an application developed by the Norwegian Institute for Nature Research (NINA). This application uses SeaBee WMSs as base layers, allowing users to identify bird locations and tag them with information describing the species, age and activity (nesting, standing, flying etc.) of each individual.

Habitat mapping annotation is conducted by the Norwegian Institute for Water Research (NIVA) and initially utilizes the Image Analyst extension for ArcGIS Pro. NIVA ecologists draw polygons to define the extent of different marine vegetation species and tag each polygon with descriptive information. Since Image Analyst supports hierarchical annotation, users can assign labels at species level where possible, or use coarser aggregations (e.g. genus) when species identification is ambiguous. Vector annotations from Image Analyst are uploaded to SeaBee storage via MinIO and processed on the JupyterHub to create standardised geopackages. These geopackages are then used by NR to train new ML models. Details of the annotation workflow for habitats are available in the SeaBee documentation.

5.5 Data publishing, sharing and visualisation

SeaBee data products are made available online using GeoServer and GeoNode. GeoServer is an open-source software server written in Java that allows users to share and edit geospatial data, while GeoNode is an open-source content management system for web GIS.

Data products from the orthorectification and ML workflows (orthophotos, habitat maps, seabird counts etc.) are first uploaded to GeoServer, where they become available as Web Mapping Services (WMSs) and Web Feature Services (WFSs). The GeoNode consumes these services and publishes them to a public web GIS interface where they can be searched and explored. The default GeoNode interface is shown in *Figure 7* and presents users with a list of SeaBee layers that can be explored interactively, together with basic tools for searching and filtering based on metadata attributes.

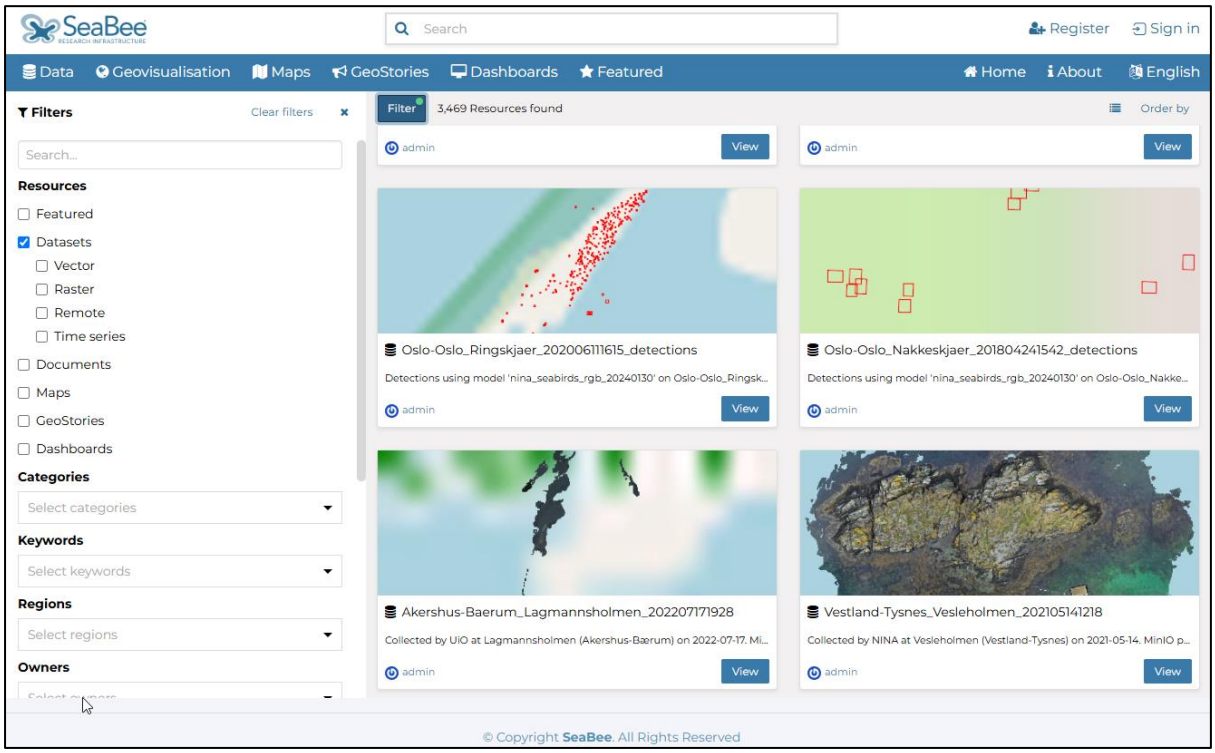

Figure 7: Standard GeoNode interface showing available data layers.

To enhance the user experience, the SeaBee data platform also offers a custom web GIS interface (*Figure 8*) built on top of GeoNode using GeoDjango, a Python framework for geospatial web applications. The main application window shows an interactive map of SeaBee missions, while a sidebar lists missions within the current view extent and provides tools for searching based on mission type, date range, and more.

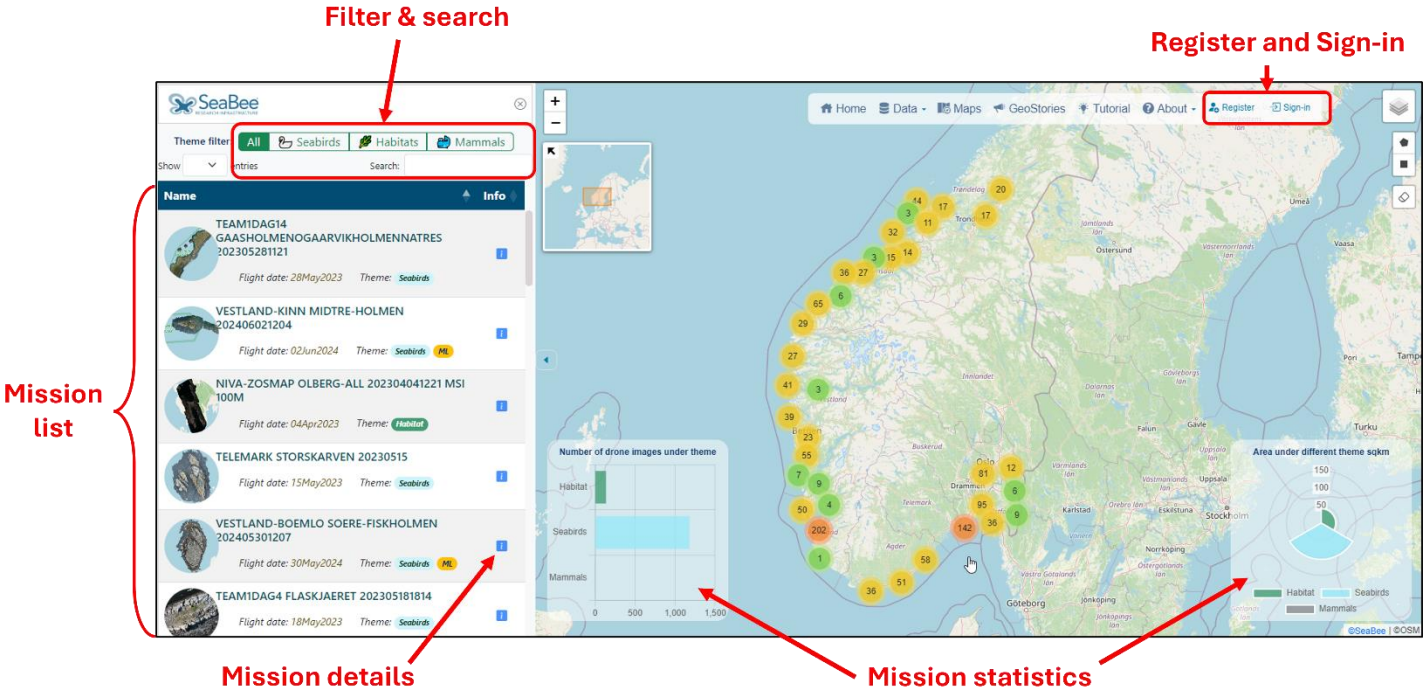

Figure 8: Main components of the SeaBee data exploration application.

Clicking on a mission in the sidebar displays a popup with additional details and options (*Figure 9*). Users can zoom to the location of the mission and overlay the orthophoto and ML results (if available) on the basemap for more detailed investigation. Registered users can also download the orthophoto and ML results to their local machine for further analysis using desktop GIS. Additionally, buttons are provided to display the WMS parameters for the mission on GeoServer and to create a direct, shareable link if desired.

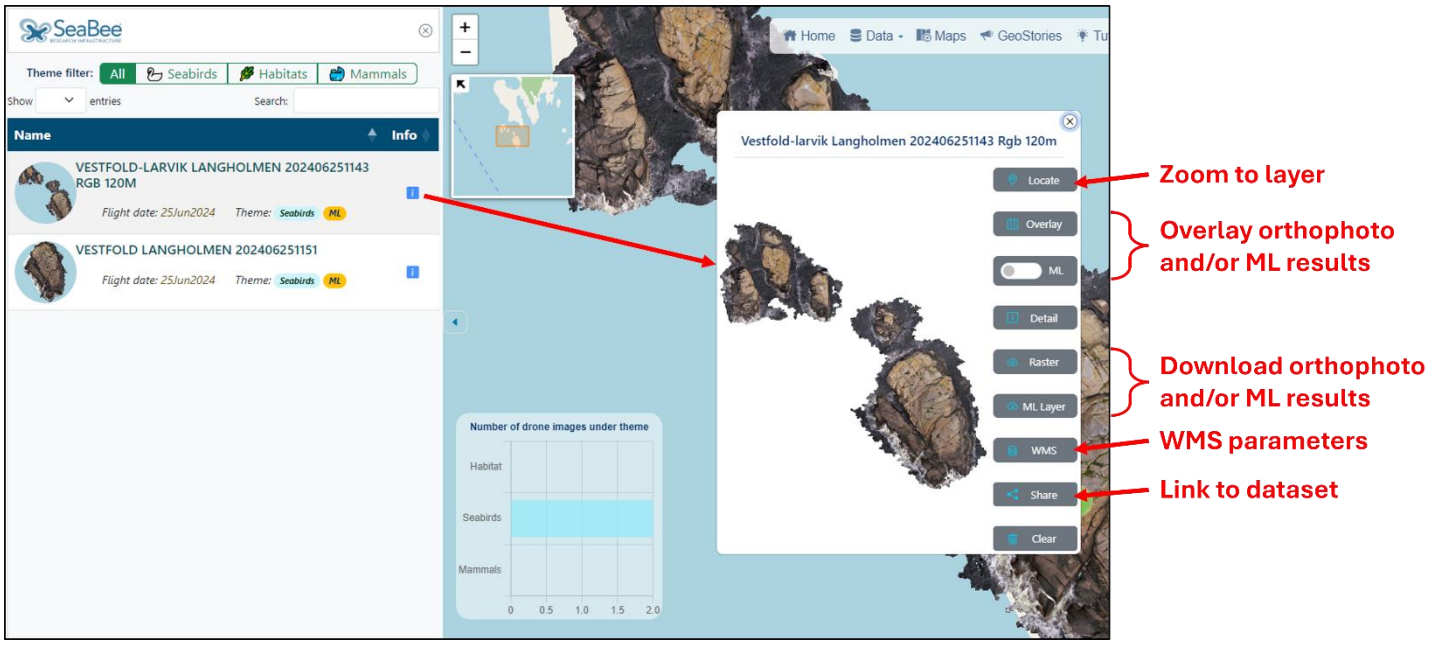

Figure 9: Mission details popup within the SeaBee data exploration application.

Registered users can use GeoNode to create Maps and GeoStories. GeoNode Maps are simple web GIS applications where users can create custom data visualisations by combining multiple SeaBee layers and other external datasets. Users can control layer styling and transparency, and enable various tools for dataset comparison, such as layer toggles and sliders. Maps can be saved to a user's GeoNode account and shared with collaborators, if desired.

GeoStories (the open-source equivalent of ESRI's StoryMaps) allow users to combine text, interactive maps, and other multimedia (images, videos etc.) to create interactive articles and reports. Like GeoNode Maps, GeoStories are saved to users' accounts and can be shared with others or made public on the platform.

5.6 JupyterHub and Python packages

JupyterHub is an open-source platform primarily designed to allow multiple users to access the JupyterLab IDE on a shared server, such as cloud infrastructure. However, JupyterHub can also deploy other web-based IDEs, such as R-Studio Server and Code Server (a browser-based implementation of VS Code).

The SeaBee platform leverages JupyterHub to provide advanced users with a comprehensive development environment customised for remote sensing and geospatial analysis. Users of the Hub can choose

between JupyterLab, R-Studio Server or Code Server according to their preferences, or switch between them for different tasks. JupyterLab is familiar to data scientists working with Python, whereas R-Studio is generally preferred by SeaBee ecologists working with R. Code Server/VS Code is often the choice for software engineers seeking a more comprehensive development environment.

Regardless of the chosen IDE, users have access to an identical file system and computational resources. The SeaBee platform offers three different JupyterHubs with varying specifications:

- **Standard.** *1 CPU, 1 GB memory*. The default machine, suitable for basic data analysis, annotation, simple statistics, and visualisation. Also suitable for deploying orthorectification tasks to ODM.
- **High memory.** *4 CPUs, 32 GB memory*. Suitable for raster processing, such as manipulating image mosaics.
- **GPU-enabled.** *4 CPUs, 16 GB memory, 1 Tesla V100-PCIE-16GB GPU*. Suitable for machine learning.

Because data structures and file paths are identical across Hubs, users can switch between machines with different resource allocations as needed. This allows workflows to be developed on smaller machines and then scaled up to larger machines once tested.

All three JupyterHubs provide users with access to an Ubuntu (Linux) machine and a consistent, customised computing environment for Python, R and Julia. Most SeaBee workflows are coded in Python, but a comprehensive set of R packages is also included for ecologists analysing SeaBee datasets directly on the platform. Julia is not extensively used at present but has potential for future use.

The JupyterHub's Python environment includes all the packages necessary to build data pipelines connecting the different "workflows" illustrated on *Figure 2*. This includes standard geospatial libraries installed at the level of the operating system (GDAL/OGR, PROJ, etc.) along with their corresponding Python bindings (rasterio/fiona, pyproj, etc.). Additionally, the environment includes three Python packages developed specifically for SeaBee:

- **seabeepy:** The main Python package supporting workflows on the SeaBee data platform. It comprises five modules, each supporting one of the main workflows illustrated on *Figure 2*:
    - The '`storage`' module provides functions for connecting to SeaBee's MinIO storage and transferring data programmatically (Section 5.1 and *Figure 3b*).
    - The '`ortho`' module contains functions associated with the orthorectification workflow, including those needed to submit tasks to NodeODM (Section 5.2).
    - The '`ml`' module contains functions for interacting with NR's machine learning API (see below and Section 5.3).
    - The '`anno`' module contains functions for standardising annotation from ArcGIS Pro's Image Analyst and building geopackages for training ML models (Section 5.4).
    - The '`geo`' module contains functions for standardising geospatial data products created by other workflows and publishing them to GeoServer and GeoNode (Section 5.5).
- **seabee-ai:** A Python package created by NR to streamline the process of deploying ML models. It provides a command line tool with an API for running NR's models with SeaBee datasets.

- **gref4hsi:** A Python package developed by NTNU for georeferencing of push-broom hyperspectral imagery, including ray-intersection, orthorectification, and co-registration. It is a core component of the pipeline for processing HSI data.

6 Automated data pipelines

6.1 Overview of pipelines

At present, there are three automated data pipelines running on the SeaBee data platform:

- The orthorectification pipeline for RGB and MSI data.
- The georeferencing pipeline for HSI data.
- The image classification pipeline for RGB and MSI orthophotos.

All three pipelines run every hour. The HSI pipeline is independent of the others, whereas the RGB and MSI pipeline for image classification runs sequentially after the orthorectification pipeline has completed. If any pipeline takes longer than one hour, the next run will not start until the previous one has finished (or failed with some error). Output from each hourly pipeline run is written to log files on MinIO. These are kept for 24 hours and then overwritten.

6.2 Pipeline development cycle

SeaBee data pipelines are initially developed by advanced users or platform administrators working on the JupyterHub. Since the software environment for pipelines is identical to that on JupyterHub, code developed and tested on the Hub can be seamlessly deployed to run on a schedule as a pipeline.

JupyterHub users have the flexibility to develop pipelines in any manner they prefer. However, so far, pipelines have primarily been developed and tested as Jupyter notebooks. These notebooks use SeaBee-specific Python packages like seabeepy, seabee-ai and gref4hsi (Section 5.6) to move data between the main SeaBee "workflows" (*Figure 2*) and perform intermediate processing.

Each notebook begins by scanning the file system for new datasets and reading settings from a standard plain text configuration file called '`config.seabee.yaml`'. Notebooks are developed and tested until running them manually via JupyterHub performs the desired processing reliably and efficiently. Once the developer is satisfied, the latest changes are pushed to GitHub and the notebook added to a list of notebooks scheduled to be executed every hour by the pipeline.

Each pipeline run starts by cloning the latest versions of the notebooks and core packages, such as seabeepy, from GitHub. This ensures that pipelines are always running the most up-to-date code and makes it easy to update pipelines simply by modifying the relevant notebook and pushing changes to GitHub.

6.3 Specification for data upload

6.3.1. Data structure

SeaBee's automated processing pipelines require raw data to be uploaded in a standard format. Data from each mission must be gathered into a single folder containing a pre-defined subfolder structure and a plain text configuration file called '`config.seabee.yaml`'. The platform does not enforce strict requirements for naming mission folders, because different pilots and organisations prefer to use their own conventions, but the contents of each mission folder must be standardised.

For RGB and MSI missions, the required subfolder structure is as follows:

```
unique_mission_folder_name/
├─ dem/
├─ gcp/
├─ ground-truth/
├─ images/
├─ orthophoto/
├─ other/
├─ report/
├─ texturing/
config.seabee.yaml
```

Note that the only mandatory components are the '`images`' subfolder, which contains the raw images, and '`config.seabee.yaml`'. A typical example of data uploaded for a RGB or MSI mission is therefore:

```
unique_mission_folder_name/
├─ images/
│  ├─ img_001.jpg
│  ├─ img_002.jpg
│  ├─ img_003.jpg
config.seabee.yaml
```

The purpose of each subfolder or file is as follows:

- **`config.seabee.yaml`** (required). A file containing flight metadata and additional settings to control the processing workflow (Section 6.3.2).
- **`images`** (required). Images from a single flight i.e. images that can be orthorectified to produce a single mosaic.
- **`dem`** (optional). Contains elevation datasets generated during orthorectification (DSMs and DTMs). This folder and its contents are generated automatically if orthorectification is performed on the platform using ODM. However, it can also be provided explicitly if orthorectification has been done elsewhere (e.g. manually on a local machine).
- **`gcp`** (optional). Folder containing ground control points (GCPs) in a standard text format. Most SeaBee missions are flown using Real Time Kinematic (RTK) positioning, in which case GCPs are not usually necessary and the folder can be omitted.
- **`ground_truth`** (optional). Ground truth data collected by field ecologists using traditional survey methods, if available.

- **`orthophoto`** (optional). Georeferenced mosaic images. Typically a single, multi-band GeoTiff with standardised band order and metadata. This folder and its contents are generated automatically if orthorectification is performed on the platform using ODM. However, it can also be provided explicitly if orthorectification has been done elsewhere (e.g. manually on a local machine).
- **`report`** (optional). Folder containing the PDF report describing results from the orthorectification process, plus any associated logs (e.g. text files and JSON). This folder and its contents are generated automatically if orthorectification is performed on the platform using ODM. However, it can also be provided explicitly if orthorectification has been done elsewhere (e.g. manually on a local machine).
- **`texturing`** (optional). Texture models generated during orthorectification. This folder and its contents are generated automatically if orthorectification is performed on the platform using ODM. However, it can also be provided explicitly if orthorectification has been done elsewhere (e.g. manually on a local machine).
- **`other`** (optional). Anything not included in the other folders.

6.3.2. Configuration file

Each mission folder must contain a file named '`config.seabee.yaml`', which contains flight metadata and settings/commands to control data processing. A minimal example with the 9 mandatory attributes is shown in *Figure 10*.

Most of the mandatory attributes provide basic mission metadata, such as the `datetime`, responsible `organisation` and SeaBee `theme` (seabirds, habitats etc.). In addition, the attributes `mosaic`, `classify` and `publish` are Boolean values that allow pilots to control which parts of the processing pipeline run on their data.

The `nfiles` attribute is mandatory because it is used by the platform to determine whether raw data upload has completed successfully before starting any further processing. This is especially important when bulk-uploading data for large numbers of missions directly from the field, where network connectivity can be patchy. Before attempting to mosaic any images using ODM, the pipeline will first check that the number of files in the `images` subfolder matches the `nfiles` attribute in the configuration file. If it does, it is assumed that upload is complete and the flight is ready to be processed; if it does not, the flight is skipped and checked again later.

```
grouping: gronningen                # General identifier linking this flight to related flights
area: agder                         # Name of specific location
datetime: '202305230830'            # 'yyyymmddHHMM' or 'yyyymmdd'. Note that the quotes are required
nfiles: 290                         # Total number of files in the 'images' folder
organisation: NINA                  # Responsible organisation
mosaic: true                        # Whether to mosaic the raw images using ODM (true or false)
classify: true                      # Whether to classify the orthomosaic using machine learning (true or false)
publish: true                       # Whether to publish the orthophoto to GeoNode (true or false)
theme: Seabirds                     # SeaBee "theme" ('Seabirds', 'Mammals', 'Habitat' or 'Water quality')
```

Figure 10: Mandatory attributes in the 'config.seabee.yaml' configuration file.

In addition to the mandatory attributes, `config.seabee.yaml` can contain optional information to control subsequent processing and add extra metadata. A complete list of currently supported attributes is shown in *Figure 11*. Pilots can use the configuration file to pass settings to ODM for use during

orthorectification, or to specify the version of the ML model used for image classification. It is also possible specify the data licence to use when results are published to the GeoNode (if `publish` is `true`).

```yaml
grouping: gronningen                                  # General identifier linking this flight to related flights
area: agder                                           # Name of specific location
datetime: '202305230830'                              # 'yyyymmddHHMM' or 'yyyymmdd'. Note that quotes are required
nfiles: 290                                           # Total number of files in the 'images' folder
organisation: NINA                                    # Responsible organisation
mosaic: true                                          # Whether to mosaic the raw images using ODM (true or false)
classify: true                                        # Whether to classify the orthomosaic using machine learning (true or false)
publish: true                                         # Whether to publish the orthophoto to GeoNode (true or false)
theme: Seabirds                                       # SeaBee "theme" ('Seabirds', 'Mammals' or 'Habitat')
spectrum_type: RGB                                    # [Optional]. Sensor type ('RGB', 'MSI', 'HSI', 'Thermal')
elevation: 40                                         # [Optional]. Flight elevation (integer > 0 in metres)
creator_name: Sindre Molværsmyr                       # [Optional]. Data collector/pilot
project: Seabirds2023                                 # [Optional]. Name of project
vehicle: DJI-M350                                     # [Optional]. Name of vehicle/drone
sensor: DJI-P1                                        # [Optional]. Name of camera/sensor
licence: CC-BY-4                                      # [Optional]. Name of licence
licence_link: https://creativecommons.org/licenses/by/4.0/  # [Optional]. Link to licence details
odm_options:                                          # [Optional]. Overrides default orthorectification settings
  dsm: true                                           # [Optional]. true or false
  dtm: true                                           # [Optional]. true or false
  cog: true                                           # [Optional]. true or false
  orthophoto-compression: LZW                         # [Optional]. JPEG, LZW, PACKBITS, DEFLATE, LZMA or NONE
  orthophoto-resolution: 0.1                          # [Optional]. Float. cm/pixel
  dem-resolution: 0.1                                 # [Optional]. Float. cm/pixel
  max-concurrency: 16                                 # [Optional]. Int
  auto-boundary: true                                 # [Optional]. true or false
  use-3dmesh: true                                    # [Optional]. true or false
  fast-orthophoto: false                              # [Optional]. true or false
  pc-rectify: false                                   # [Optional]. Bool
  split: 999999                                       # [Optional]. Int
  split-overlap: 150                                  # [Optional]. Int
  crop: 3                                             # [Optional]. Int or Float (>= 0)
  pc-quality: high                                    # [Optional]. ultra, high, medium, low, lowest
  feature-quality: high                               # [Optional]. ultra, high, medium, low, lowest
  radiometric-calibration: camera                     # [Optional]. 'camera', 'camera+sun' or None
ml_options:                                           # [Optional]. Overrides default machine learning settings
  task:                                               # [Optional]. detection, segmentation
  model:                                              # [Optional]. Name/version of machine learning model to use
```

Figure 11: Complete list of attributes supported by 'config.seabee.yaml'.

If optional settings are omitted from the configuration file, default values are used instead. By default, the platform will use the latest version of the ML model, and settings for ODM that are designed to achieve adequate results over a broad range of mission types. Unless otherwise specified, all SeaBee datasets are published under a CC-BY-4 Creative Commons licence, which permits sharing and adapting the data as long as attribution is provided (including for commercial purposes).

7 Platform statistics

As of March 2025, the SeaBee platform hosts approximately 100 TB of active data stored on NIRD's Data Peak service (with the same amount backed up to the Data Lake). These data come from 1857 aerial drone missions, of which 1572 are from seabird surveys (covering ~114 km^2), 254 are habitat mapping (covering ~95 km^2), and 31 are for assessing water quality (covering ~0.1 km^2). Most of the datasets so far have been generated by the RGB and MSI data pipelines, but usage of the HSI pipeline has increased considerably during recent months. It is anticipated that the annual seabird survey conducted by NINA in late spring 2025 will generate another 30 to 50 TB of data.

8 Future development

The SeaBee data platform is continuously evolving, with new components added regularly. Recently, SeaBee acquired a green Light Detection and Ranging (LiDAR) sensor, and preliminary work has begun to develop a LiDAR data pipeline, including 3D visualization of categorised point clouds. Additionally, a second workflow is under development to implement "direct georeferencing" of RGB and MSI imagery. Direct georeferencing aims to geolocate individual images without the need for computationally demanding orthorectification (e.g. using ODM). Although direct georeferencing is currently less accurate than traditional orthorectification, traditional methods often fail over open water due to the difficulty of identifying common features across multiple images. As drone Inertial Measurement Units (IMUs) improve, direct georeferencing is becoming more feasible. A robust pipeline for direct georeferencing will enable SeaBee to explore areas further offshore, which is essential for surveying large marine mammal species, such as whales.

9 Conclusion

The SeaBee data platform is fully operational, providing Norwegian researchers with efficient and scalable processing of data collected by flying drones. It offers automated workflows for processing data from RGB, multispectral, and hyperspectral sensors, creating an end-to-end pipeline from raw data upload to online publishing of data products, including orthophotos and machine learning-based image classifications.

SeaBee data products adhere to the FAIR principles (Findable, Accessible, Interoperable, and Reusable) and are accessible via the SeaBee geoportal. This portal allows users to search and explore SeaBee data, download processed datasets, and develop their own applications using SeaBee data. Additionally, the platform provides powerful tools for advanced users to analyse SeaBee data in the cloud using Python or R.

We believe the SeaBee platform is the first of its kind and it has already proven its value by streamlining several large-scale ecological surveys, including the annual seabird population studies conducted by NINA. We hope SeaBee will become a vital component of Norway's national research infrastructure and significantly contribute to sustainable management of the coastal zone.

The Norwegian Institute for Water Research

We are Norway's premier research institute in the fields of water and the environment. We are experts on ecosystems in both freshwater and marine environments, from mountains, lakes and rivers, to fjords, coasts and oceans. We develop science-based knowledge and solutions to challenges related to the interaction between water and climate, the environment, nature, people, resources and society.