PFG (2023) 91:157–170
https://doi.org/10.1007/s41064-023-00242-2

ORIGINAL ARTICLE

Semantic Real-Time Mapping with UAVs

Phillipp Fanta-Jende[1] · Daniel Steininger[1] · Alexander Kern[2] · Verena Widhalm[1] · Javier G. Apud Baca[1] · Markus Hofstätter[1] · Julia Simon[1] · Felix Bruckmüller[1] · Christoph Sulzbachner[1]

Received: 16 December 2022 / Accepted: 13 April 2023 / Published online: 11 May 2023

Abstract
Whilst mapping with UAVs has become an established tool for geodata acquisition in many domains, certain time-critical applications, such as crisis and disaster response, demand fast geodata processing pipelines rather than photogrammetric post-processing approaches. Based on our 3D-capable real-time mapping pipeline, this contribution presents not only an array of optimisations of the original implementation but also an extension towards understanding the image content with respect to land cover and object detection using machine learning. This paper (1) describes the pipeline in its entirety, (2) compares the performance of the semantic labelling and object detection models quantitatively and (3) showcases real-world experiments with qualitative evaluations.

Keywords UAV · Real-time mapping · Semantic labelling · Object detection · vSLAM

1 Introduction

Unmanned aerial vehicles (UAVs) have become an essential asset for acquiring geoinformation for various purposes and in many domains. UAVs have democratised geodata acquisition by being affordable, versatile and offering ease-of-use. Due to this role as a low-threshold technology, UAVs did not only enter the consumer market quickly but achieved a high maturity in a short amount of time. This makes their use attractive but also feasible for risky and difficult applications with a high demand for reliable and robust solutions, such as crisis and disaster management. Here, UAVs can serve as a support tool for decision-making of first responders relying on quick data capture, processing and eventually (automated) interpretation (Erdelj et al. 2017). UAV photogrammetry as a whole can be considered a mature technology but lacks real-time performance. Disaster relief, however, requires timely data availability—ideally in real-time. To make this possible, an array of technologies and methods needs to work hand in hand, from industrial imaging sensors, navigation instruments and high-bandwidth communication links to real-time capable state estimation and reconstruction algorithms as well as object detection and semantic labelling models. This contribution summarises our efforts towards a real-time capable UAV mapping solution with direct data interpretation.

Phillipp Fanta-Jende, Daniel Steininger and Alexander Kern have contributed equally to this work.

✉ Phillipp Fanta-Jende
phillipp.fanta-jende@ait.ac.at

Daniel Steininger
daniel.steininger@ait.ac.at

Alexander Kern
a.kern@tu-braunschweig.de

Verena Widhalm
verena.widhalm@ait.ac.at

Javier G. Apud Baca
Javier.Apud-Baca@ait.ac.at

Markus Hofstätter
markus.hofstaetter@ait.ac.at

Julia Simon
julia.simon@ait.ac.at

Felix Bruckmüller
felix.bruckmueller@ait.ac.at

Christoph Sulzbachner
christoph.sulzbachner@ait.ac.at

[1] Center for Vision, Automation & Control, AIT Austrian Institute of Technology GmbH, Giefinggasse 4, Vienna 1220, Austria

[2] Institute of Flight Guidance, Technische Universität Braunschweig, Universitätsplatz 2, 38106 Braunschweig, Lower Saxony, Germany

2 Related Work

Our research work intersects various fields, most prominently represented by efforts to achieve rapid geometric transformations of image data and their respective interpretation. To summarise these major research directions, this literature review discusses rapid mapping approaches and machine learning in this context separately.

2.1 Rapid Mapping

Creating orthomosaics of an area whilst the acquiring UAV is still in the air is a challenging problem combining techniques from geodesy, robotics and computer vision as it involves self-localisation of a moving camera, dense scene reconstruction in real-time whilst fulfilling geographical accuracy constraints. Early approaches simplified the underlying assumptions by applying traditional image stitching techniques as described by Szeliski (2007). There, image alignment is treated as a 2D problem computing the affine transformation or homography between consecutive frames based on feature extraction and matching. This strategy neglects surface elevation, but is computationally efficient and was utilised for aerial map creation by various authors (Botterill et al 2010; Kekec et al. 2014). The results of such frameworks, however, are erroneous especially in low altitudes. Distortions in the image induced by perspective variations accumulate and lead to drift in the global mosaic. Additionally, no 3D surface information is extracted which highly benefits a variety of use cases, such as situational awareness in a disaster relief scenario.

Consequently, research has been carried out over the last decades to overcome these limitations and allow for the generation of true orthomosaics in real-time. Hein et al. (2019) use a priori Digital Elevation Models (DEMs) to account for surface elevation. However, availability of accurate and up-to-date DEMs is limited and even completely infeasible when the observed scene drastically changed (due to, e.g. an earthquake). Bu et al. (2016) utilise a visual SLAM for camera pose estimation and project the images into a common reference frame to create the global mosaic. This does not account for distortions through surface elevation, yet it is more robust in low altitudes by modelling the true 3D camera motion. Hinzmann et al. (2018) were the first to our knowledge to combine both the pose estimation and the dense scene reconstruction into a real-time framework. By multi-sensor fusion of an inertial measurement unit (IMU), a Global Navigation Satellite receiver (GNSS) and optical flow estimates of a KLT-tracker, the resulting pose and image was fed into a stereo reconstruction pipeline. The surface was recovered through computation and fusion of depth maps for the creation of an elevation map to eventually rectify the image data. Based on this work, another pipeline for orthomosaicing was developed called OpenREALM (Kern et al. 2020a). It modularises the design of Hinzmann et al., so that each individual processing stage can be computed either in the air or on the ground. Additionally, the sensor fusion was replaced by an extendable visual SLAM interface, so that state of the art implementations [e.g. ORB SLAM2 (Mur-Artal and Tardós 2017)] could be used for camera pose estimation. Since then, other publications have pushed the state of the art even further. Zhao et al. (2021) tackled the problem of low overlapping image sequences as it is often the case in aerial imagery through tighter incorporation of GNSS data into the underlying visual SLAM. A similar strategy was followed by Miller et al. (2022) who employ a pose graph optimisation backend to fuse visual and GNSS measurements. In addition, each new keyframe of their pipeline is passed through a semantic segmentation framework to assign basic labels such as road, vegetation or building to ground segments. Combining aerial data with scene understanding is also a key target of our research within this paper.

2.2 Machine Learning

Recent advances in artificial intelligence and graphics-accelerated parallel computing lead to faster and more efficient machine learning techniques. These techniques can be applied in a broad spectrum of fields, including computer vision (Lin et al. 2014). In this realm, Convolutional Neural Networks (CNNs) (Krizhevsky et al. 2017; Yu et al. 2017) demonstrated to be an outstanding method for inferring semantic information and object detection in RGB images with a relatively low time taxation. In contrast, more recently emerging techniques like Vision Transformers (Zheng et al. 2021; Dosovitskiy et al. 2020) achieve promising results for classification problems but not without a higher time expense. Performance and low time consumption make CNNs more desirable for our real-time mapping use case. Going into detail, CNN architectures for object detection are divided into one- and two-stage approaches with the former expediting real-time performance (Zou et al. 2019) at the expense of extra accuracy. Given our use case, we focus on one-stage detectors. One of the most popular was first introduced in 2016 as YOLO (Redmon et al. 2016). Low computational complexity and relatively high accuracy quickly established YOLO as one of the go-to one-step detectors. Since then, newer iterations further improved the model's performance. The current model iteration is YOLOv7, published in 2022 (Wang et al. 2022).

Remote sensing applications represent a challenging domain for all the data-driven machine learning algorithms, since the available case-specific training data are commonly scarce. For instance, the nadir view for our case scenario does not match the vast majority of publicly available machine learning datasets. The current limitations for the tasks of semantic labelling and object detection are discussed in the following sections.

2.2.1 Semantic Labelling

In contrast to popular all-purpose (Lin et al. 2014; Kuznetsova et al. 2020) or driving (Cordts et al. 2016; Neuhold et al. 2017) datasets, which typically offer a front facing view, remote sensing datasets usually employ a nadir perspective.

A handful of datasets with semantic labelling annotations compatible with our use case became available in recent years. For instance, FloodNet (Rahnemoonfar et al. 2021), which has gained a lot of popularity recently, consists of almost 0.4 k images and according pixel-wise annotations. Similarly, SemanticDrone (TU Graz (ICG) 2022) consists of 0.4 k images captured at an altitude of 5–30 m above ground and is especially relevant due to the high number of included scene participants. However, since most images were recorded over a small settlement of newly built prefabricated houses, the low variability of the included data limits its generalisation to more challenging scenarios. Closely, LandCoverAI (Boguszewski et al. 2021) provides 41 orthorectified high-resolution aerial images captured in Poland. Similarly, Ruralscapes (Marcu et al. 2020) includes 20 UAV-based videos depicting a typical rural setting with 1 k manually annotated images with segmentation masks.

2.2.2 Object Detection

Front-facing camera object detection datasets are common. Examples like Pascal VOC (Everingham et al. 2010) or MS COCO (Lin et al. 2014) contain relevant categories such as pedestrians or motorised vehicles. However, the perspectives from which images were acquired in these cases are not compatible with the mapping application. In comparison, the DOTA dataset (Xia et al. 2018) is based on satellite and aerial images from Google Earth. Its latest version includes 11 k images and nearly 1800 k instances of 18 common categories. Nonetheless, the dataset does not contain annotations of small classes such as pedestrians. The most suitable dataset for our use case is VisDrone (Du et al. 2019), which comprises 8.6 k images captured by multiple drone types in 14 Chinese cities containing different weather, lighting and environmental conditions. Additionally, the recordings show a compromise between top and front viewing angles at different flight altitudes.

3 Methodology

Two complementary development lines were followed: First, implementation of a pipeline capable of real-time mapping, semantic segmentation and object detection. Second, the establishment of a benchmark dataset to measure the performance of the machine learning models on our use case and the corresponding dataset selection for model training.

3.1 Semantic Orthorectification Pipeline

The pipeline is implemented as a modular series of downstream stages or nodes, each one receiving, processing and forwarding a proprietary Robot Operating System (ROS) message named *Frame*. A Frame message contains the information generated at each step (including data capture) and facilitates communication between nodes. At its core, the mapping task is based on OpenREALM (Kern et al. 2020a, 2021), which was first published as an open source project on Github (Kern et al. 2020b). In comparison to the original, this version features a separation into air and ground modules. The former is comprised of the data acquisition and pose-estimation nodes and runs on the target UAV. The latter includes the remaining nodes, some of which were modified to achieve real-time performance and runs on a separate computer, referred to as ground station.

Communication between the air and ground modules is achieved using an LTE wireless connection router. A detailed description of the hardware is available in Sect. 4.1.1.

The complete pipeline comprises eight distinctive processing stages in the following order: data acquisition 3.1.1, pose estimation 3.1.2, densification 3.1.3, surface generation 3.1.4, segmentation 3.1.5, orthorectification 3.1.6, tileing 3.1.7 and object detection 3.1.8. The workflow of the pipeline is depicted in Fig. 1.

Additionally, a Geoserver node was added for data visualisation and further use of the data products in a GIS 3.2. However, this does not impact the performance of any of the previous nodes.

3.1.1 Data Acquisition

Readings are gathered from the available sensors on the UAV and encapsulated in our proprietary Frame message for easy transportation and data control. In general, this interface is agnostic to the mode of input, e.g. ROS topics/messages (as in our case) or a folder containing graphic formats such as JPG files with EXIF headers both work as

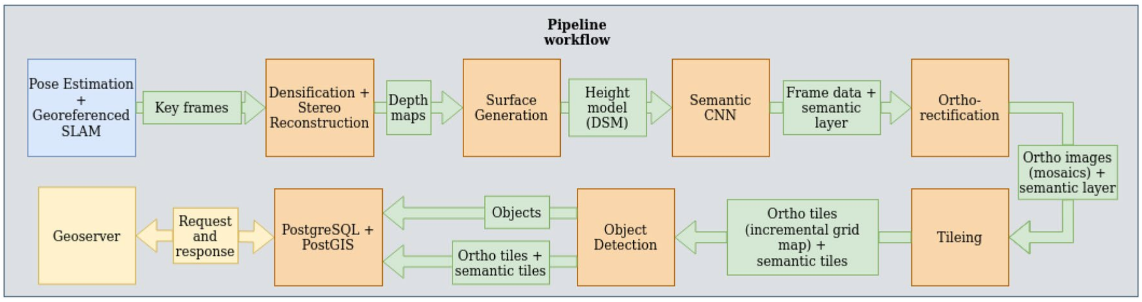
Pipeline workflow
Pose Estimation + Georeferenced SLAM
Key frames
Densification + Stereo Reconstruction
Depth maps
Surface Generation
Height model (DSM)
Semantic CNN
Frame data + semantic layer
Ortho-rectification
Ortho images (mosaics) + semantic layer
Tileing
Ortho tiles (incremental grid map) + semantic tiles
Object Detection
Objects
Ortho tiles + semantic tiles
PostgreSQL + PostGIS
Request and response
Geoserver

Fig. 1 Pipeline workflow overview. The blue block refers to the UAV onboard node. Orange blocks outline the ground station pipeline nodes. Yellow blocks refer to the Geoserver processes. Most relevant data transferred from node to node is shown in green blocks

long as the required image topic and its respective intrinsic and extrinsic orientation elements can be parsed.

3.1.2 Pose Estimation

Camera pose estimation is essential to determine the correct position of each image at a global scale. Accuracy is essential in this node as inaccuracies cannot be compensated afterwards. To support the state estimation, the UAV image stream is used in conjunction with a visual SLAM approach. To this end, a vSLAM interface was implemented to integrate different frameworks, for instance ORB SLAM2 (Mur-Artal and Tardós 2017), which for the most part follows the traditional structure of feature extraction, matching and optimisation through bundle adjustment. Because monocular cameras are not able to recover the scene scale without additional information, the next step is to align the vSLAM trajectory with measurements of other sensors. By combining GNSS information with the relative altitude of the onboard barometer, we can extract a 3D trajectory with a metric scale. The similarity transformation between the two trajectories, the metric and visual one, can be extracted through least-squares estimation. Whilst the procedure is straightforward, it comes with some disadvantages. The estimation neglects the actual sensor orientation and assumes the position of the GNSS receiver and the camera's focal point to be approximately the same. Consequently, a minimum trajectory length is required for reasons of robustness and to estimate the azimuth angle. In our framework, the trajectory alignment is performed iteratively for every keyframe and once the estimation error converges below a certain threshold, the computed similarity transformation is used as a georeference for all frames and the pipeline initialisation is complete.

Only the georeferenced RGB keyframes and the corresponding acquisition information (UAV heading, speed, coordinates, etc.) are forwarded to the ground station to execute the following pipeline steps as a Frame message using the wireless connection. The exclusive use of keyframes provides enough information for a good reconstruction whilst reducing the bandwidth requirement between the UAV and the ground station.

Although modifiable, in our use case, the vSLAM is operating at 15 frames per second as this is the ideal compromise between onboard processing capabilities and reconstruction quality.

3.1.3 Densification

After collecting at least two overlapping frames viewed from different viewpoints, depth maps are computed. By observing each respective 3D point, only those whose estimated position is consistent across multiple depth maps are flagged as valid. As a result of this stage, each Frame carries a 3D estimate of its observed scene. Similar to the pose estimation stage, external frameworks for stereo reconstruction can be incorporated via an interface. The presented pipeline utilises the Plane Sweep Library (Häne et al. 2014).

3.1.4 Surface Generation

The filtered depth maps then enter the surface generation node. By projecting the depth maps into 3D space, a point cloud of the currently observed scene is reconstructed. To reduce computational costs in the following steps, a georeferenced grid map is created with its cells assigned an elevation value depending on the nearest points in the point cloud. Potential holes in the grid are filled through interpolation. Consequently, each Frame now holds a watertight update to the global digital surface model (DSM).

3.1.5 Semantic Labelling

Semantic segmentation for terrain labelling is applied to the RGB information in the georeferenced Frame at this point, since the best results can be achieved on full-resolution keyframes prior to rectification. Models are specifically trained for the target application, as described in Sect. 3.3.2 and deployed by converting them from PyTorch to the Open Neural Network Exchange (ONNX) standard (Bai et al. 2019). ONNX is a platform-independent open source format used to achieve suitable performance for the pipeline using optimisation tools and different compilers. The resulting segmentation masks are then embedded in the Frame message forwarded to subsequent nodes, allowing for an efficient transformation of the labels along with the RGB data throughout the remaining pipeline in real-time.

3.1.6 Orthorectification

The surface information is utilised to remove perspective distortions from the original RGB image data as well as the semantic layer. To this end, a backprojection from grid proposed by Hinzmann et al. (2018) was implemented. By iterating through each cell of the DSM and projecting its 3D point into the camera plane, perspective distortions in the RGB data are minimised and orthogonality is enforced. The corrected data are added to the grid map as an additional layer resulting in an incremental, multi-layered grid update for the global mosaic. The same rectification transformations are subsequently applied to the segmentation mask.

3.1.7 Tileing

Excluding the pose estimation, all previous stages can be considered stateless. Consequently, the resources per Frame only depend on the current input data and there is no increase in resource requirements over time. This, however, does not hold true during visualisation. The growing number of rectified mosaics generated during mapping represents an increasing load on memory when visualising the map. During long sessions, this resource is prone to run out. Moreover, this factor may also represent an obstacle when performing a real-time reconstruction.

Hence, a resource-efficient and real-time tileing stage that combines two concepts is implemented. First, in a similar fashion to the Tiled Map Service (Open Source Geospatial Foundation 2022d) for tiled web maps, the global map is treated as a grid and the incoming map is tiled to fill in the cells accordingly. The incremental grid maps from the orthorectification node are divided into 256 × 256 pixels tiles and individually georeferenced to address the geographical coordinates shift that arises from the division (tiles further away from the upper left corner of the mosaic have a coordinate shift correction equivalent to 256[px]*GSD[px/m]*tile shift). If a smaller portion of the mosaic does not totally cover the corresponding tile area, the missing pixels are filled in by setting the transparency to 100%, as shown in Fig. 2. In this way, there is no interference when a partial tile overlaps a full tile. Tiles are then fused with the corresponding cell content (if already existing) or first initialised in the global grid map and stored to disc (Fig. 3). Tileing runs for as long as the mapping process does, meaning that areas can be revisited and remapped during the same flight. This is achieved with continuous caching, which increases the workload on the CPU, but enables creating and displaying maps whose size is only limited by the available hard drive storage in real-time. Concerning applications, such as cooperative multi-UAV mapping, this would theoretically allow for covering large areas in high resolution in a matter of minutes with fixed-wing UAVs. Tileing reduces the memory requirements when displaying as only the necessary tiles are loaded.

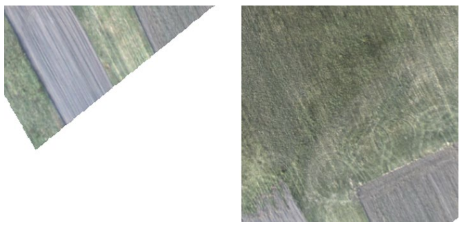
Fig. 2 Examples of incomplete and full tiles

Second, most web tile systems use the WGS 84-based Pseudo-Mercator coordinate system (EPSG 3857) (MapTiler 2022) for index-based tile georeferencing. However, the pipeline uses UTM and does not rely on index-based referencing based on PNG files but writes TIF files with corresponding headers (Open Source Geospatial Foundation 2022b). Combining these two approaches decreases memory consumption whilst displaying the map and eradicates the need of intermediate coordinate transformation, resulting in a reduced overall tileing time of around 100 ms. Moreover, resorting to TIF files enables high compatibility to existing geodata infrastructures.

3.1.8 Object Detection

Object detection is performed on the final RGB tiles using the specifically trained CNN described in Sect. 3.3.3. To obtain a trained model with increased robustness against image transformations, data augmentation techniques are used. For example, mosaic augmentation (Bochkovskiy et al. 2020), which creates a synthetic input image consisting of four randomly transformed images of the training data. Since object detection requires more complex pre-processing and post-processing of images than semantic labelling, the corresponding processing stage builds upon the TensorRT framework for deployment to achieve real-time performance. The resulting bounding boxes and object labels are georeferenced based on the input tile's information and integrated into the final map.

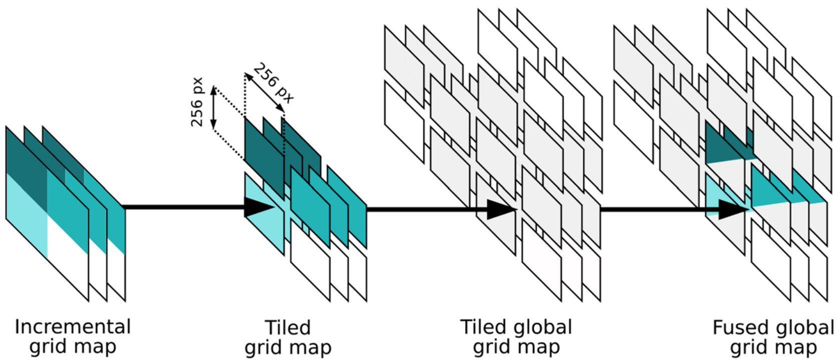

Fig. 3 Incremental grid maps generated by the orthorectification stage consist of several layers, e.g. the rectified RGB data or the observed surface elevation. The grid maps are sliced into individual tiles. These multi-layered tiles are fused with the existing data in the global mosaic and then saved to the hard drive

3.2 Server Segment and Data Infrastructure

The resulting georeferenced tiles with the highest resolution (i.e. up to 6.5 cm ground sample distance at 100 m altitude with the camera used in our setup described in Sect. 4.1.1) are collected in a geospatial data management server based on GeoServer, an open source server that enables geospatial data sharing, processing and editing with support for PostGIS (PostGIS 2022) as a spatial database extension for an object-relational PostgreSQL (The PostgreSQL Global Development Group 2022) database. The tiles are processed using the GeoServer's image mosaic plugin (Open Source Geospatial Foundation 2022c) allowing for the creation of composite images from a set of georeferenced rasters with one query. The database is organised to process the raster file location, geospatial information, time information and flight mission to enable the processing of time series of flights and missions. To ensure the real-time visualisation capability of the GIS, new tiles are inserted directly into the PostgreSQL database to ensure the highest performance. To query composite images from the server, a Web Map Service (WMS) with a Common Query Language (CQL) extension (Open Source Geospatial Foundation 2022a) is used to filter the bounding box, time interval and flight mission. This further minimises the computational performance for the server to ensure real-time capability.

3.3 Machine Learning

Multiple experiments were performed to produce suitable CNN-based models for semantic terrain labelling and object detection, the best performing of which are integrated in the corresponding pipeline stages. The following sections provide details regarding our custom benchmark dataset and the experimental setup.

3.3.1 Benchmark Dataset

Quantifying the performance of machine learning tasks for the intended application scenarios and sensor setups requires specific test data. As the open source datasets show strong differences in their acquisition altitude, the benchmark dataset is divided into three different altitudes with different image scales. Multiple sequences are recorded with the target UAV platform at altitudes of 80, 100 and 120 m to measure their impact on model performance and show their degree of generalisation over multiple resolution levels. From these recordings, we manually sampled a representative set of 273 images which sufficiently cover all altitudes and relevant classes. Each sample is manually annotated with per-pixel masks, differentiating between the semantic categories of *Grass*, *Tree*, *Building*, *Road*, *Vehicle* and *Person*, as visualised in Fig. 4. To evaluate object detection, corresponding bounding boxes for the latter two categories are automatically derived. Due to the characteristics of the captured area and intended application scenario, the vegetation classes are significantly more frequent than man-made

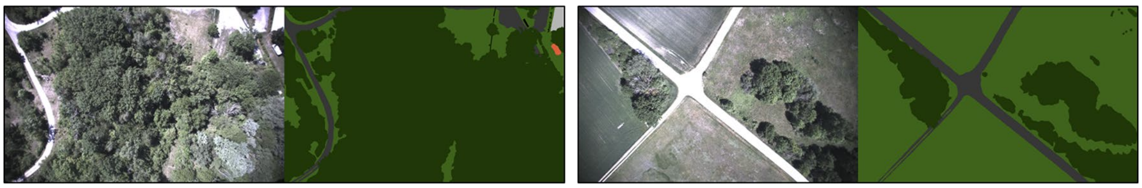

Fig. 4 Representative pairs of images and semantic masks annotated in the benchmark dataset containing the categories *Grass* (light green), *Tree* (dark green), *Building* (light grey), *Road* (dark grey) and *Vehicle* (orange)

Table 1 Listing of all target benchmark categories and their corresponding category denominations in the mapped source datasets

Category	Id	LandCoverAI	FloodNet	SemanticDrone
Grass	0	Background	Grass	Dirt
				Grass
Tree	1	Woodland	Tree	Vegetation
				Tree
				Bald tree
Building	2	Building		Roof
			Building flooded	Wall
			Building non-flooded	Window
				Door
Road	3	Road	Road non-flooded	Paved area
				Gravel

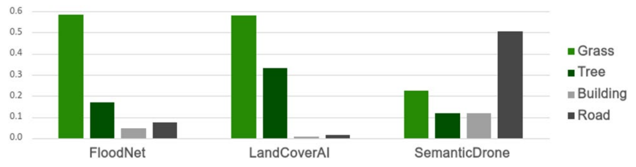

Fig. 5 Normalised label distribution of source datasets

structures, whilst vehicles and persons are underrepresented. Whilst these data gaps can be addressed when extending the benchmark dataset for future applications, the data still provide relevant insights, since it was recorded with the target platform and sensor setup.

3.3.2 Semantic Labelling

The segmentation models are trained using multiple publicly available datasets. Of those discussed in Sect. 2.2.1, we selected FloodNet, SemanticDrone and LandCoverAI due their recording altitudes and class distributions roughly resembling the given application scenario. Prior to training and evaluation, the labels of each source dataset are mapped to the benchmark categories of *Grass*, *Tree*, *Building* and *Road*, as listed in Table 1. Unused labels are excluded for training and validation. The resulting label distribution of each training dataset after mapping is visualised in Fig. 5. Whilst the labels *Gras* and *Trees* mainly appear in FloodNet and LandCoverAI, they have a lack of *Building* and *Road* instances compared to the SemanticDrone dataset which contains 50% *Road* and over 10% *Building* areas. LandCoverAI has an image size of 9000×9500 and 4200×4700 pixels with a per-pixel resolution of 25 cm and 50 cm GSD. To make the dataset comparable to the others and adapt it to our use case, the ortho images are split into pieces with a resulting image size of 1280×1280 pixels, whilst FloodNet and SemanticDrone images are downscaled to 1920×1280 pixels to achieve a similar per-pixel resolution.

For all experiments, we apply a split between training and validation data of 90:10 to the source dataset and test on the entire benchmark dataset. During training, each sample is randomly resized and cropped to a final input size of 640×640 pixels to meet performance and memory constraints, whilst validation and test images are constantly scaled to 1920×1280 pixels. We selected the efficient architecture dla-34 (Yu et al. 2018) to facilitate real-time processing on the target platform. All models are trained from scratch on a system containing two NVIDIA 3090 RTX GPUs using a learning rate of 0.001 with a step size of 50 and a batch size of 20. The best-performing models are selected based on validation results after 200 epochs. For quantitative evaluations, we use the established intersection-over-union (IoU) metric (Everingham et al. 2010).

3.3.3 Object Detection

Since small foreground classes, such as vehicles and persons, are typically highly underrepresented in aerial segmentation datasets, their robust identification requires the additional task of object detection. After analysing several available datasets, we selected MS COCO (Lin et al. 2014) and VisDrone (Du et al. 2019), since both provide a sufficient amount of relevant samples captured from various altitudes resulting in different GSDs and especially the latter one partially provides nadir views. The source labels are aggregated into two target classes, one for pedestrians and one for motorised and non-motorised vehicles. We quantify the results using the average precision (AP) metric as defined by the COCO benchmark (Lin et al. 2014), which is more challenging than those used by Pascal VOC (Everingham et al. 2010), since it represents averaged results over multiple IoU thresholds. For training an object detector, we used the framework of YOLOv5 (Jocher et al. 2020), as it performs better than YOLOv6. Whilst the newest release, YOLOv7, would provide slightly superior performance, the selected version offers the best trade-off between accuracy, computational complexity and training time. All experiments are conducted by fine-tuning a model pre-trained on all categories of the COCO dataset to the relevant target classes *Vehicle* and *Person*. Models are trained on the same hardware described in Sect. 3.3.2 using the default training configuration and standard data augmentation techniques, such as random scaling, mosaic augmentation and cropping to focus on tiny objects up to 10×10 pixels. We use the network architecture YOLOv5m of release 6.0 with an input size of 1280×1280 pixels and a batch size of 8. We select the best-performing model after 37 epochs for further evaluations.

4 Experiments

Although the artificial intelligence models and the orthorectification pipeline are integrated into in a single pipeline, each module can be evaluated separately. As a first step, benchmark experiments are conducted for creating suitable models for semantic labelling and object detection. After integrating them into the corresponding modules, the entire pipeline is evaluated in a real-time setup.

4.1 Semantic Orthorectification Pipeline Experiments

4.1.1 Setup and Data Acquisition

Flight experimentation is divided into real-time (online) and post-processing (offline). Online implies a split pipeline, i.e. the data-acquisition and pose-estimation nodes are running on an UAV, whilst the remaining nodes run on the ground station. The connection is established using a wireless telecommunication node. In comparison, offline means that the UAV recorded data, later to be retrieved and fed into the pipeline running entirely on the ground station. The main differences between these approaches are the real-time execution in online mode and the higher computing capacity as well as bandwidth in offline mode. The densification node is disabled for the experiments.

In both cases, flights are performed using a fixed-wing Skywalker EVE-2000 UAV (2240 mm wingspan, 1270 mm fuselage length and 4600 g maximum take-off mass) equipped with a Pixhawk 2 as Flight Management System (FMS) and running the Arduplane flight controller stack in autonomous configuration (Fig. 6). GNSS measurements for flight control and pose estimation are captured using a Here3 from Cube Pilot with real-time kinematics (RTK) enabled. Flight recordings and UAV pipeline nodes run on a NVIDIA Jetson Xavier NX. Imagery was acquired using a 2.3 megapixel global shutter CMOS colour camera with a resolution of 1920 × 1200 pixels based on a Sony IMX249 sensor mounted on a cardanic suspension system for roll compensation. Images are compressed before sending and processing.

Communication is addressed by establishing a private connection over the local 4G network using a Teltonika RUT240 LTE router on the UAV to a Teltonika RUTX11 LTE router on the ground station. This configuration grants a good bandwidth, which translates to a stable connection and no substantial amount of lag observed during experimentation. Alternatively, the setup can be operated with an autonomous tracking antenna system running a Wi-Fi network (802.11n) or proprietary datalink to be independent of local telecommunication infrastructure.

The ground station features a dual 10-core XEON processor and four NVIDIA RTX 2080 TI graphic cards. The station acts as a control station to send/receive information to/from the UAV and process the acquired data (pipeline and machine learning). A total of three flights were executed for testing and/or recording. Each followed the same flight path with a fixed height of 100 m to correspond with the target scenario (large coverage at minimal time) and avoid motion blur and consequent tracking loss at lower altitudes due to the relatively high speed above ground of around 75 km/h. (please note that this height refers to the data acquisition only, not the benchmark dataset recordings):

- Flight 1: Exclusive data recording for offline experiments.
- Flight 2: Data recording plus online rapid mapping (split pipeline running) and no artificial intelligence nodes.
- Flight 3: Exclusive online rapid mapping and artificial intelligence nodes running for semantic segmentation and object detection.

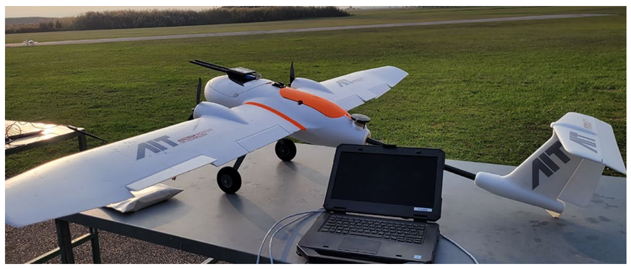
Fig. 6 Fixed-wing UAV used in the experiments

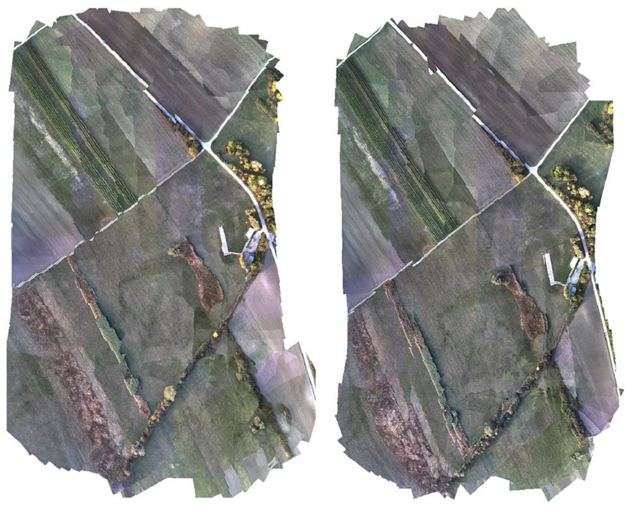
Fig. 7 Online vs offline map reconstruction (Flight 2)

Table 2 Relative accuracy between orthomosaics of Flight 1 (offline), Flight 2 (offline and online) and Flight 3 (online) with Pix4D orthomosaic as reference

	RMSE X [m]	RMSE Y [m]	Mean X [m]	Mean Y [m]	Min error [m]	Max error [m]
Flight 1 offline	1.03	1.97	0.47	−0.39	0.08	5.09
Flight 2 offline	2.39	5.19	−1.51	−3.84	0.06	8.66
Flight 2 online	4.07	3.47	−0.05	0.44	0.08	8.88
Flight 3 online	3.92	3.86	−2.17	−0.19	0.01	9.28

4.1.2 Qualitative Analysis of Reconstructed Data Products and Real-Time Capabilities

As visible in Figs. 7 and 8, the online flight (left) is more prone to high frequency tile shifting or misalignment compared to the offline approach (right). This effect, however, is not limited to well-defined structures like roads. Tile shifting is also visible in flat surfaces as colour changes and seam lines on smooth surfaces. In spite of the more significant tile shifting visible in the online setup, the whole reconstruction took place in real-time.

Previous experiments (Kern et al. 2021) focussed on understanding the impact of different SLAM approaches [ORB-SLAM3 (Campos et al. 2021), OV2SLAM (Ferrera et al. 2021), OpenVSLAM (Sumikura et al. 2019)] on the reconstruction quality as well as on the absolute accuracy. In our past experiment, a total of 27 ground control points were surveyed to determine the RSME in X, Y and Z. Depending on the acquisition altitude and the underlying SLAM approach, absolute accuracies of around 1 m in the horizontal and vertical dimension were achieved. To ascertain the difference in accuracy between offline and online processing, 20 checkpoints were measured in a reference orthomosaic processed using Pix4D without the use of ground control (Table 2).

Besides a visually more consistent reconstruction (Figs. 7 right and 8 right), the offline reconstruction also yields a more accurate data product overall. However, Flight 1 is the only flight without an active pose reconstruction node running in parallel entailing that all the available onboard computing resources are assigned to data recording. Against this background, the decreased accuracy of Flight 2 offline can likely be attributed to performance limitations of the onboard processing unit since computing resources are distributed between recording and pose estimation. The recording node, however, does not impact the accuracy to a great degree, since the online data products of Flight 2 and Flight 3 share a similar accuracy. The error distribution (see Fig. 9) shows degrading accuracy towards the edges of the orthomosaic. Although this may be usual behaviour in traditional photogrammetric workflows due to fewer across-track image overlaps, this pattern is likely to be caused by a more difficult tracking scenario for the underlying SLAM. In particular, repetitive patterns, as they occur in fields and meadows, are difficult for feature matching approaches and

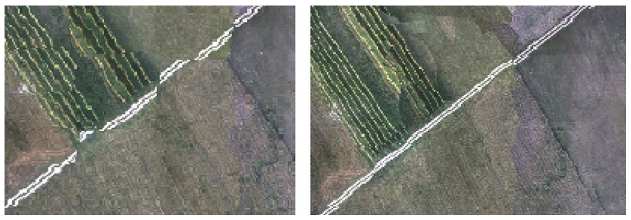
Fig. 8 Close up of online vs offline shifted tiles (Flight 2)

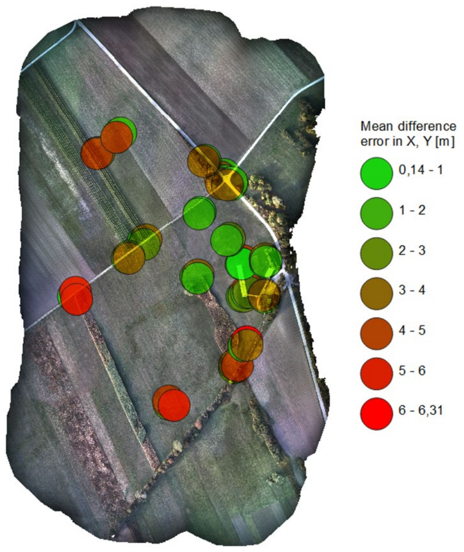

Fig. 9 Locations of checkpoints with mean difference error in X, Y

consequently affect the pose estimation, especially in non-inertial SLAM implementations.

Regarding the real-time capability, the delay between image acquisition and the reception of a keyframe on the ground station is between 1 and 2 s. On top of this, the ground station processing time increases the total processing time including tile generation to around 5–6 s.

It is expected that the frame reception on the ground station from the UAV may vary on different factors, such as telecommunication infrastructure, bandwidth usage, obstruction (e.g. trees), physical distance between sender and receiver nodes and/or pipeline buffers usage. Nonetheless, we did not experience any substantial variation during the experiments. The frequency of tile generation remained constant during the flights.

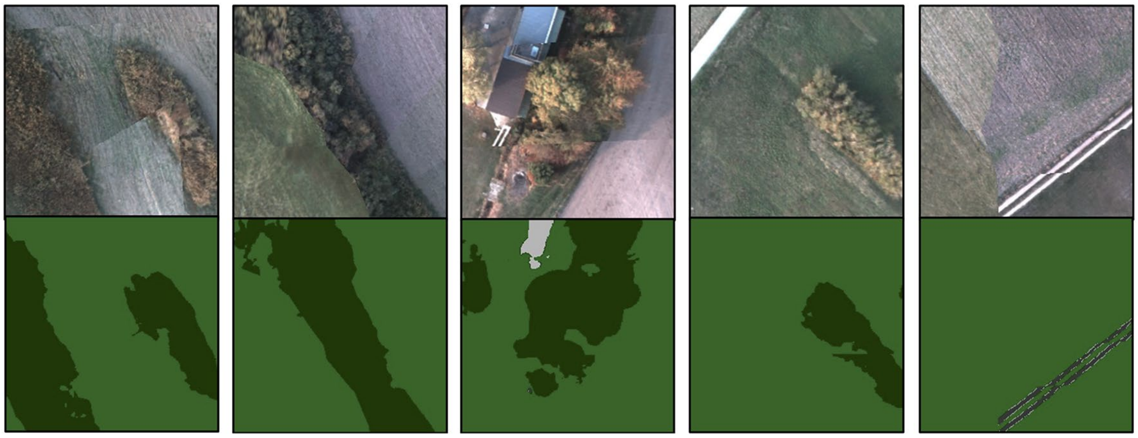

Fig. 10 Representative tiled results of online semantic labelling (bottom row) for the categories *Grass* (light green), *Tree* (dark green), *Building* (light grey) and *Road* (dark grey), along with corresponding RGB tiles (top row)

4.1.3 Qualitative Analysis of Semantic Labelling and Object Detection

Evaluating the online experiments confirmed that the semantic labelling module provides sufficient performance for real-time application of the entire pipeline on the target platform. Figure 10 shows a selection of the results as orthorectified tiles. They demonstrate that the segmentation results are largely consistent across overlapping input frames, as there are hardly any label switches along borders between stitched source images.

Apart from the integration of the semantic labelling module into our pipeline to process unaltered RGB images at an early stage, the real-time evaluation also shows the model's generalisation capability to the application scenario. Especially, the classes *Grass* and *Tree* can be distinguished with high accuracy despite varying visual appearance, as visible in first two images from the left in Fig. 10. Buildings are more challenging and more scarce in the test data, but still correctly assigned in many cases. Performance for roads strongly depends on their visual appearance as visible in the two images on the right, where overexposed or underexposed instances are often missed, whilst those resembling the training data are usually correctly classified. Overall, the results resemble those from offline tests on the benchmark data, indicating that the model is suitably integrated in the processing pipeline and generalises well to the changes in environmental conditions and camera settings between the recordings. A visualisation of the complete segmented map of Flight 2 is shown in Fig. 12.

The object detection module is evaluated on the processed tiles for the *Vehicles* category, since the flight altitude (100 m) and resulting resolution (0.15 m/px) are considered insufficient for robustly detecting persons from a nadir view. The cars present in the evaluated scenarios are consistently

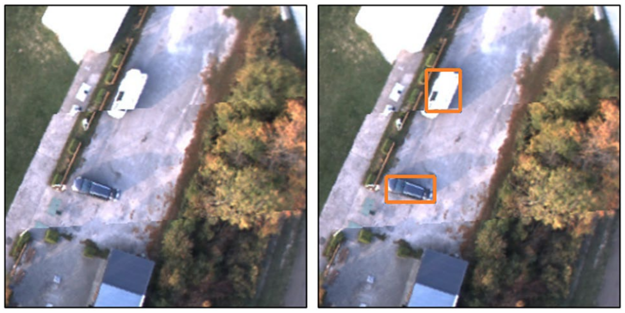

Fig. 11 On the left image, the input image tile is shown, whereby the image on the right side shows the representative object detection results on the tiling output for the *Vehicle* category (orange)

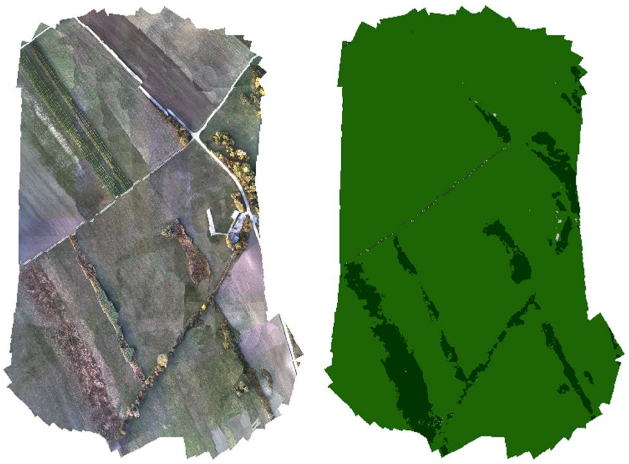

Fig. 12 Visualisation of Flight 2 reconstructed RGB map (left) and segmentation map generated from key frames (right)

Table 3 Segmentation performance as IoU per category for each training dataset on the corresponding validation sets

	Grass	Tree	Road	Building	Overall
FloodNet	70.7	45.3	40.2	43.1	49.8
LandCoverAI	79.5	72.1	54.1	39.9	61.4
SemanticDrone	79.0	65.6	51.4	85.0	70.2

detected, even if they coincide with tileing artefacts, as visible in Fig. 11. Furthermore, the model did not produce any false detections across the entire recording and proved to provide real-time performance on the target platform.

4.2 Benchmark Experiments

To create suitable models for the intended application scenario, we conducted multiple experiments for both validation sets corresponding to the selected training data and our custom evaluation benchmark. The quantitative results for the tasks of semantic labelling and object detection are presented in the following subsections.

4.2.1 Semantic Labelling

The three models described in Sect. 3.3.2 were evaluated both on their own respective validation sets and the benchmark dataset defined in Sect. 3.3.1, as summarised in Tables 3 and 4. The former results do not provide a measure for the overall performance of models, but rather indicate how challenging each source dataset is for the given semantic labelling tasks. Generalisation capability can then be derived by comparing the results with those on the benchmark dataset. Although the measured performance decrease of SemanticDrone and FloodNet in overall IoU is only 2.0 and 5.5, respectively, LandCoverAI outperforms them by a large margin in most categories. All models are highly robust against changes in altitude, which only marginally impacts performance. The relatively low performance of SemanticDrone is not surprising, since the image data have a stronger bias towards man-made structures and therefore only a small overlap to the given application scenario.

Based on the quantitative results, we decided to use the model trained on LandCoverAI for further evaluation and integration in our rapid mapping pipeline. A selection of qualitative results is shown in Fig. 13.

4.2.2 Object Detection

The two detection models trained on the VisDrone and MS COCO datasets reach overall AP values of 42.6% and 52.7% on their corresponding validation sets. However, applying the same models to our benchmark dataset reverses the ranking with 70% and 59.7%, respectively, for the *Vehicle* class. Whilst the MS COCO model performs better on its own validation set, the advantages of the VisDrone model are

Table 4 Segmentation performance as IoU per category and flight altitude for each trained model on the benchmark dataset

	Grass	Tree	Road	Building	80 m	100 m	120 m	Overall
FloodNet	85.0	**39.1**	5.9	3.3	52.1	49.8	49.8	50.9
LandCoverAI	**91.0**	36.1	**23.4**	**12.7**	**54.7**	**56.1**	**56.2**	**55.4**
SemanticDrone	19.4	22.6	4.3	2.5	13.3	17.9	15.1	15.1

The highest IoUs are highlighted for each category

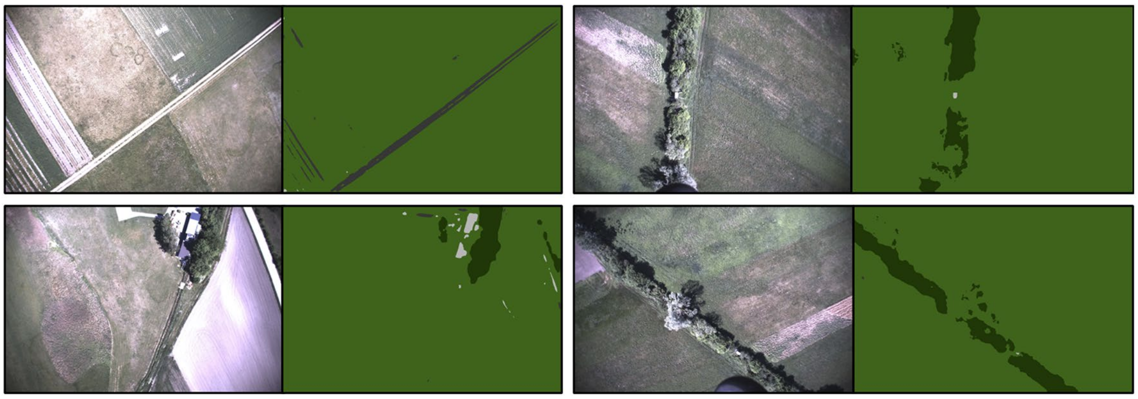

Fig. 13 Representative segmentation results of the model trained on LandCoverAI on the benchmark dataset for the categories *Grass* (light green), *Tree* (dark green), *Building* (light grey) and *Road* (dark grey)

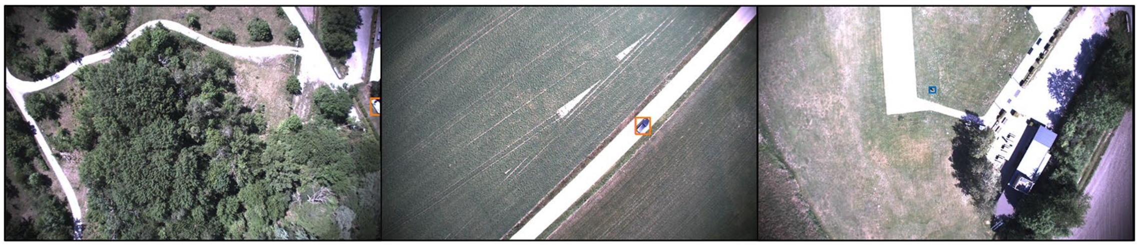

Fig. 14 Representative detection results on the benchmark dataset of the model trained on the VisDrone dataset for the categories *Vehicle* (orange) and *Person* (blue). On the right image, a detected person is visible in the centre of the image and the other two images show a single prediction of a parking and a driving vehicle

shown on the benchmark data with more than 10% increased AP due to the higher inherent variability of vehicles captured from non-canonical views. The class *Person* is evaluated only qualitatively due to a lack of annotated persons within the benchmark data. Based on the results, the model trained on the VisDrone dataset is used for the final pipeline. In Fig. 14, three examples of predictions from the resulting model for both classes are shown.

5 Discussion

Our experiments show that real-time mapping with AI-based scene understanding is feasible. In a disaster scenario, our target scenario for this setup, fast coverage is the most important factor for geodata collection. Our system maps and interprets a scene in real-time offering first results within the first 5–6 s after acquisition and covering an area of 45 ha in about 7.5 min. To achieve this, we employed a fixed-wing UAV during experimentation, allowing for higher speed, longer lasting flights in comparison to average copter UAVs. This, however, necessitates a higher camera frame rate to avoid tracking loss due to smaller image overlaps. Consequently, more powerful computing capabilities on the UAV and higher bandwidth for communication to ground become mandatory. In this setup, most hardware and software components can be swapped and adapted to the respective use case. For instance, the semantic labelling and object detection nodes can be efficiently extended towards other categories depending on the scenario (e.g. forest fires, floods, landslides), whilst images of a thermal camera can be rectified alongside the RGB input. By combining object detection and semantic labelling, small classes such as vehicles and persons can be identified in addition to larger semantic regions. Our future developments will target multiple system components. First, the reconstruction quality and, consequently, the overall accuracy will most likely benefit from a pose estimation based on visual-inertial SLAM. This will not only stabilise the estimated trajectory but also mitigate tracking loss. Second, although the pipeline allows for multi-UAV setups, true collaborative mapping will involve a joint reconstruction pipeline for further improving the reconstruction accuracy. Third, extending the AI nodes to other classes and objects will facilitate the application in other (disaster) scenarios. This will be achieved by adding new datasets from future releases as well as creating and developing our own dataset, in particular for flooding scenarios.

6 Conclusion

This contribution outlines the methodology as well as the implementation and application of a real-time orthomosaicing pipeline with AI-based terrain segmentation and domain-specific object detection. Our quantitative and qualitative experiments on reconstruction and model quality demonstrated the applicability of our system for rapid mapping tasks, in particular within the framework of time-critical applications, such as disaster response. Especially in the fields of photogrammetry and remote sensing, real-time capability comes at the expense of accuracy. To this end, future iterations of the pipeline will involve optimisations of pose estimation, an extension towards multi-UAV joint reconstruction and the inclusion of other classes and objects in the AI models.

Acknowledgements This research has been funded by the Austrian security research programme KIRAS of the Federal Ministry of Finance (BMF). In addition, we would like to thank Marlene Glawischnig and Vanessa Klugsberger, who manually annotated the benchmark dataset.

Funding Open access funding provided by AIT Austrian Institute of Technology GmbH.

Data availability The data used for this paper cannot be released publicly. We, however, published the majority of the source code of our pipeline (see GitHub reference).

Declarations

Conflict of Interest The authors declare that they have no conflict of interest.

Code Availability Rapid mapping code to be available in https://github.com/laxnpander/OpenREALM.

References

Bai J, Lu F, Zhang K et al (2019) Onnx: Open neural network exchange. https://github.com/onnx/onnx

Bochkovskiy A, Wang CY, Liao HYM (2020) Yolov4: optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934

Boguszewski A, Batorski D, Ziemba-Jankowska N et al (2021) Landcover. ai: Dataset for automatic mapping of buildings, woodlands, water and roads from aerial imagery. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 1102–1110. https://doi.org/10.1109/CVPRW53098.2021.00121

Botterill T, Mills S, Green R (2010) Real-time aerial image mosaicing. Int Conf Image Vis Comput New Zealand. https://doi.org/10.1109/IVCNZ.2010.6148850

Bu S, Zhao Y, Wan G et al (2016) Map2dfusion: Real-time incremental uav image mosaicing based on monocular slam. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 4564–4571. https://doi.org/10.1109/IROS.2016.7759672

Campos C, Elvira R, Rodrıguez JJG et al (2021) Orb-slam3: an accurate open-source library for visual, visual-inertial, and multimap slam. IEEE Trans Rob 37(6):1874–1890. https://doi.org/10.1109/TRO.2021.3075644

Cordts M, Omran M, Ramos S et al (2016) The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp 3213–3223. https://doi.org/10.1109/CVPR.2016.350

Dosovitskiy A, Beyer L, Kolesnikov A et al (2020) An image is worth 16x16 words: transformers for image recognition at scale. arXiv preprint https://doi.org/10.48550/arXiv.2010.11929

Du D, Zhu P, Wen L et al (2019) Visdrone-det2019: The vision meets drone object detection in image challenge results. In: Proceedings of the IEEE/CVF international conference on computer vision workshops. https://doi.org/10.1109/ICCVW.2019.00030

Erdelj M, Natalizio E, Chowdhury KR et al (2017) Help from the sky: leveraging uavs for disaster management. IEEE Pervasive Comput 16(1):24–32. https://doi.org/10.1109/MPRV.2017.11

Everingham M, Van Gool L, Williams CK et al (2010) The pascal visual object classes (voc) challenge. Int J Comput Vis 88(2):303–338. https://doi.org/10.1007/s11263-009-0275-4

Ferrera M, Eudes A, Moras J et al (2021) Ovslam: a fully online and versatile visual slam for real-time applications. IEEE Robot Autom Lett 6(2):1399–1406. https://doi.org/10.48550/arXiv.2102.04060

Häne C, Heng L, Lee GH et al (2014) Real-time direct dense matching on fisheye images using plane-sweeping stereo. In: 2014 2nd International Conference on 3D Vision, pp 57–64. https://doi.org/10.1109/3DV.2014.77

Hein D, Kraft T, Brauchle J et al (2019) Integrated uav-based real-time mapping for security applications. ISPRS Int J Geo Inf 8(5):219. https://doi.org/10.3390/ijgi8050219

Hinzmann T, Schönberger JL, Pollefeys M et al (2018) Mapping on the fly: real-time 3d dense reconstruction, digital surface map and incremental orthomosaic generation for unmanned aerial vehicles. In: Hutter M, Siegwart R (eds) Field Service Robot. Springer International Publishing, Cham, pp 383–396. https://doi.org/10.1007/978-3-319-67361-5_25

Jocher G, Nishimura K, Mineeva T et al (2020) yolov5. https://github.com/ultralytics/yolov5

Kekec T, Yildirim A, Unel M (2014) A new approach to real-time mosaicing of aerial images. Robot Auton Syst 62(12):1755–1767. https://doi.org/10.1016/j.robot.2014.07.010

Kern A, Fanta-Jende P, Glira P et al (2021) An accurate real-time Uav mapping solution for the generation of Orthomosaics and surface models. ISPRS - Int Arch Photogramm Remote Sens Spatial Inf Sci 43B1:165–171. https://doi.org/10.5194/isprs-archives-XLIII-B1-2021-165-2021

Kern A, Bobbe M, Khedar Y et al (2020a) Openrealm: real-time mapping for unmanned aerial vehicles. In: 2020 International Conference on Unmanned Aircraft Systems (ICUAS), pp 902–911. https://doi.org/10.1109/ICUAS48674.2020.9213960

Kern A, Bobbe M, Khedar Y et al (2020b) Openrealm: real-time mapping for unmanned aerial vehicles. In: 2020 International Conference on Unmanned Aircraft Systems (ICUAS), pp 902–911. https://doi.org/10.1109/ICUAS48674.2020.9213960

Krizhevsky A, Sutskever I, Hinton GE (2017) Imagenet classification with deep convolutional neural networks. Commun ACM 60(6):84–90. https://doi.org/10.1145/3065386

Kuznetsova A, Rom H, Alldrin N et al (2020) The open images dataset v4. Int J Comput Vis 128(7):1956–1981. https://doi.org/10.1007/s11263-020-01316-z

Lin TY, Maire M, Belongie S et al (2014) Microsoft coco: common objects in context. In: European conference on computer vision. Springer, pp 740–755. https://doi.org/10.1007/978-3-319-10602-1_48

MapTiler (2022) EPSG 3857. https://epsg.io/3857, accessed: 2022-11-20

Marcu A, Licaret V, Costea D et al (2020) Semantics through time: semi-supervised segmentation of aerial videos with iterative label propagation. In: Proceedings of the Asian Conference on Computer Vision. https://doi.org/10.1007/978-3-030-69525-5_32

Miller ID, Cladera F, Smith T et al (2022) Stronger together: air-ground robotic collaboration using semantics. IEEE Robot Autom Lett 7(4):9643–9650. https://doi.org/10.1109/LRA.2022.3191165

Mur-Artal R, Tardós JD (2017) Orb-slam2: an open-source slam system for monocular, stereo, and rgb-d cameras. IEEE Trans Rob 33(5):1255–1262. https://doi.org/10.1109/TRO.2017.2705103

Neuhold G, Ollmann T, Rota Bulo S et al (2017) The mapillary vistas dataset for semantic understanding of street scenes. In: Proceedings of the IEEE international conference on computer vision. pp 4990–4999. https://doi.org/10.1109/ICCV.2017.534

Open Source Geospatial Foundation (2022a) CQL and ECQL. https://docs.geoserver.org/stable/en/user/tutorials/cql/cql_tutorial.html, accessed: 2022-11-20

Open Source Geospatial Foundation (2022b) GDAL. https://gdal.org, accessed: 2022-11-20

Open Source Geospatial Foundation (2022c) GeoServer ImageMosaic Plugin. https://docs.geoserver.org/stable/en/user/data/raster/imagemosaic, accessed: 2022-11-20

Open Source Geospatial Foundation (2022d) Tiled map service specifications. https://wiki.osgeo.org/wiki/Tile_Map_Service_Specification, accessed: 2022-10-13

PostGIS (2022) PostGIS. https://postgis.net, accessed: 2022-11-20

Rahnemoonfar M, Chowdhury T, Sarkar A et al (2021) Floodnet: a high resolution aerial imagery dataset for post flood scene understanding. IEEE Access 9:89,644-89,654. https://doi.org/10.1109/ACCESS.2021.3090981

Redmon J, Divvala S, Girshick R et al (2016) You only look once: unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp 779–788. https://doi.org/10.1109/CVPR.2016.91

Sumikura S, Shibuya M, Sakurada K (2019) Openvslam: a versatile visual slam framework. In: Proceedings of the 27th ACM International Conference on Multimedia. pp 2292–2295. https://doi.org/10.1145/3343031.3350539

Szeliski R et al (2007) Image alignment and stitching: a tutorial. Found Trends ® Comput Graph Vis 2(1):1–104. https://doi.org/10.1561/0600000009

The PostgreSQL Global Development Group (2022) PostgreSQL. https://www.postgresql.org, accessed: 2022-11-20

TU Graz (ICG) (2022) Semantic drone dataset v1.1. https://dronedataset.icg.tugraz.at, accessed: 2022-11-20

Wang CY, Bochkovskiy A, Liao HYM (2022) Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv preprint https://doi.org/10.48550/arXiv.2207.02696

Xia GS, Bai X, Ding J et al (2018) Dota: a large-scale dataset for object detection in aerial images. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3974–3983. https://doi.org/10.1109/CVPR.2018.00418

Yu F, Koltun V, Funkhouser T (2017) Dilated residual networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 472–480. https://doi.org/10.1109/CVPR.2017.75

Yu F, Wang D, Shelhamer E, et al (2018) Deep layer aggregation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2403–2412. https://doi.org/10.1109/CVPR.2018.00255

Zhao Y, Chen L, Zhang X et al (2021) Rtsfm: real-time structure from motion for mosaicing and dsm mapping of sequential aerial images with low overlap. IEEE Trans Geosci Remote Sens 60:1–15. https://doi.org/10.1109/TGRS.2021.3090203

Zheng S, Lu J, Zhao H et al (2021) Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 6881–6890. https://doi.org/10.1109/CVPR46437.2021.00681

Zou Z, Shi Z, Guo Y et al (2019) Object detection in 20 years: a survey. arXiv preprint https://doi.org/10.48550/arXiv.1905.05055