Received: 22 October 2021 | Revised: 21 April 2022 | Accepted: 6 May 2022

DOI: 10.1002/rob.22082

RESEARCH ARTICLE

WILEY

Deep learning with RGB and thermal images onboard a drone for monitoring operations

Simon Speth[1]* | Artur Gonçalves[2] | Bastien Rigault[2] | Satoshi Suzuki[3] | Mondher Bouazizi[4] | Yutaka Matsuo[5] | Helmut Prendinger[2]

[1]Department of Informatics, Technical University of Munich, Munich, Germany

[2]Digital Content and Media Sciences Research Division, National Institute of Informatics, Tokyo, Japan

[3]Graduate School of Engineering, Chiba University, Chiba, Japan

[4]Faculty of Science and Technology, Keio University, Yokohama, Japan

[5]Department of Technology Management for Innovation (TMI), and Program for Social Innovation (PSI), Center for Engineering (RACE), School of Engineering, The University of Tokyo, Tokyo, Japan

Correspondence
Helmut Prendinger, National Institute of Informatics, Tokyo, Japan.
Email: helmut@nii.ac.jp

Funding information
Kakenhi 16H06562

Abstract

This article describes the artificial intelligence (AI) component of a drone for monitoring and patrolling tasks associated with disaster relief missions in specific restricted disaster scenarios, as specified by the Advanced Robotics Foundation in Japan. The AI component uses deep learning models for environment recognition and object detection. For environment recognition, we use semantic segmentation, or pixel-wise labeling, based on RGB images. Object detection is key for detecting and locating people in need. Since people are relatively small objects from the drone perspective, we use both RGB and thermal images. To train our models, we created a novel multispectral and publicly available data set of people. We used a geo-location method to locate people on the ground. The semantic segmentation models were extensively tested using different feature extractors. We created two dedicated data sets, which we have made publicly available. Compared with the baseline model, the best-performing model could increase the mean intersection over union (IoU) by 1.3%. Furthermore, we compared two types of person detection models. The first one is an ensemble model that combines RGB and thermal information via "late fusion"; the second one is a 4-channel model that combines these two types of information in an "early fusion" manner. The results suggest that the 4-channel model had a 40.6% increase of average precision for stricter IoU values (0.75) compared with the ensemble model and a 5.8% increase in the average precision compared with the thermal model. All models were deployed and tested on the NVIDIA AGX Xavier platform. To the best of our knowledge, this study was the first to use both RGB and thermal data from the perspective of a drone for monitoring tasks.

KEYWORDS
convolution neural networks, deep learning, disaster response, drone, thermal imaging

[Correction added on 6 July 2022, after first online publication: The URL in the dataset website has been replaced from "https://www.okutamasegmentation.org/" to "https://www.okutama-segmentation.org/".]

*A major part of this work was conducted while the author was an International Internship student at the National Institute of Informatics, Tokyo.

1 | INTRODUCTION

Japan has suffered 11 earthquakes of magnitude 7.0 or higher within the last 10 years (US Geological Survey [USGS], 2020). Hence, the Japanese government and researchers in Japan are actively exploring response strategies to provide relief during such large-scale disasters (Kitano et al., 1999; Matsuno & Tadokoro, 2004; Tadokoro, 2009). Since earthquakes are difficult to predict, it is important to prepare fast and accurate search-and-rescue efforts (Matsuno & Tadokoro, 2004). Here, "fast" means that a rescue operation is necessary within 3 h as the survival rate dramatically decreases over time (Tadokoro, 2009). For example, 80% of causalities of the 1995 Hanshin-Awaji Earthquake occurred within the first 3 h after the disaster and 94% of victims died within the first 24 h (Aoki et al., 2004).

We propose the use of unmanned aerial vehicles (UAVs), also called "drones," and artificial intelligence (AI), specifically deep learning (DL), as essential technologies for monitoring and patrolling tasks in the disaster response phase of situation awareness (Endsley, 1995). This awareness provides the basis for subsequent fast and effective decision-making during a large-scale earthquake.

Our research relates to the following tasks in a disaster relief mission. In Mission Phase 1, UAVs must scan the disaster area to detect and locate people. An even more important task is to determine and compute a ground route leading to the detected people. Therefore, we must identify roads blocked by landslides or fallen trees, as they might prevent the access of rescue cars to people in need. In Mission Phase 2, UAVs must deliver supplies to the rescuers. The UAV must fly to the location of a rescuer and find a suitable landing spot, which is (i) at a safe distance to the rescuers and (ii) not blocked by any objects. Our paper focuses on Mission Phase 1.

For practical reasons, the scenarios considered in this paper are specific and limited to monitoring and patrolling using UAVs. The Advanced Robotics Foundation (ARF)[1] in Japan identified certain disaster relief missions as important tasks. Examples are detection of road blockage by wood or landslide and detection and location of people. However, the current problem definition does not consider people covered or occluded by gravel. Appendix A provides a part of ARF's description of disaster relief missions.

The drone we used is a custom-built hexa-rotor UAV (Figure 1) with a target flight time of 25 min at a payload of 1 kg. It was developed by the Chinese company *A.Y.Drone*. The recent advancement in graphics processing unit (GPU) technology enabled us to deploy the DL components directly onto the embedded system onboard the UAV.

This approach has several advantages over sending recorded images to a data center for analysis. For example, the method can still operate if a cellular network infrastructure has been partially destroyed during a disaster. Furthermore, communication with the data center would increase network traffic in areas in which the disaster is imminent and might prevent people from contacting authorities or their families.

[1]https://arf.or.jp

FIGURE 1 Our hexa-rotor drone, the A.Y. X01, without the NVIDIA AGX Xavier onboard computer or cameras.

The mission requirements considered in our paper translate into two technical challenges, environment recognition and person detection, which we address by developing DL models for semantic segmentation and object (person) detection. In environment recognition using semantic segmentation, the practical challenge is that we do not know which objects will block the road; thus, we cannot aim to develop such a specific classifier. Additionally, we must determine suitable landing spots for the UAVs. Therefore, we developed an eight-class semantic segmentation, or pixel-wise labeling, model to perform both "road blockage" and "landing spot" detection (Figure 2). For instance, since we aim to detect road blockages, we introduced the label "paved ground." This category is also beneficial for suggesting suitable landing spots. Additional categories that aid in identifying landing spots are include "water" and "plants." However, semantic segmentation is not the preferred method for person detection from the drone perspective. Unlike "large" object categories such as "plants," "paved road," or "buildings," people consume only a relatively small number of pixels from the drone perspective; thus, they are easily missed. Furthermore, semantic segmentation only provides "blobs" of person pixels, not individual person detection, and we would still require another type of detector. Therefore, we developed a dedicated person detection method that benefits from both RGB and thermal data and a 45 ° camera angle.

These are the three main contributions of this study.

- First, we achieved new semantic segmentation results from the drone perspective by improving on the research by Laurmaa (2016). Here, we have made two dedicated data sets (Okutama Drone and Swiss Drone) publicly available at https://www.okutama-segmentation.org. We also analyzed the performance-to-latency trade-off and deployed a DL model on the NVIDIA AGX Xavier platform.
- Second, we created a novel multispectral data set of people from the drone perspective, with 6000 aligned RGB and far-infrared (FIR) image frames, containing 19,000 instances of people. The data set is publicly available at https://www.nii-cu-multispectral.org.
- Third, we achieved novel results for person detection via drones using thermal data. We compared two popular models, YOLOv3 and RetinaNet, on single-spectral person detection. Subsequently,

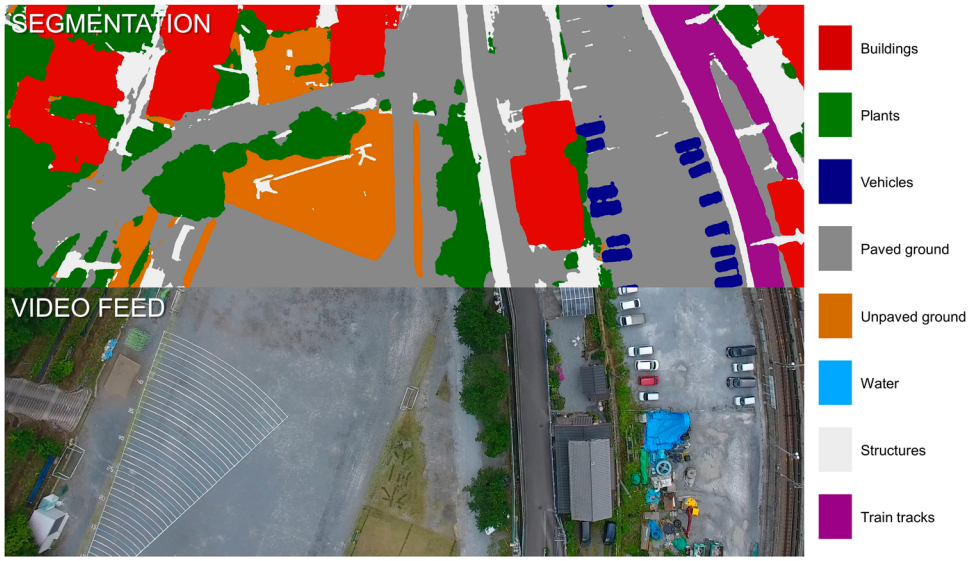

FIGURE 2 Sample frame from the semantic segmentation model. The frame was recorded over Okutama at a flight altitude of 100 m. The eight labels (classes) are explained to the right. The label "person" was not used as people were now detected using a dedicated person detection model. An animated version is available online at https://youtu.be/6ja94uOB9fs.

we designed two types of multispectral models for detecting humans using drones. The first one is an ensemble model that combines RGB and thermal information via "late fusion"; the second one is a 4-channel model, using "early fusion." For the 4-channel model, only the first layer of the standard model was replaced with a new convolutional layer. Thereby, the pre-trained weights from the original architecture were used for initializing the new convolutional layer. For the ensemble model, a fusion layer was developed in which the RGB and thermal bounding boxes were merged. All models were deployed and tested on an NVIDIA AGX Xavier platform.

The remainder of this article is organized as follows. Section 2 focuses on related studies on DL and approaches for semantic segmentation and object detection. Section 3 describes the components and architecture of our AI onboard software. Section 4 introduces our semantic segmentation model and results. Here, we also briefly explain the use of semantic segmentation to detect landing spots and road blockages. Section 5 describes the creation of a new multispectral data set for person detection from a drone perspective. Section 6 presents our novel approach of person detection with RGB and thermal images by drone. Section 6.5 describes some observations from the field. Finally, Section 7 discusses further research and concludes the paper.

2 | RELATED STUDIES

This section exclusively reports on DL approaches to vision-related tasks. Studies on disaster response using drones are discussed, for example, in Geraldes et al. (2019).

In recent years, computer vision has been revolutionized owing to the advances in deep neural networks (DNNs), which achieved unprecedented performance. Beginning with Ciresan et al. (2011) and Krizhevsky et al. (2012) (AlexNet), model performance steadily improved on data sets such as PASCAL VOC, MS COCO, and ImageNet (Everingham et al., 2010; Lin et al., 2014; Russakovsky et al., 2015).

New trends in DL for vision tasks were characterized by the substitution of multiple layers by convolutional ones, thus introducing what is referred to as convolutional neural networks (CNNs) and fully convolutional networks (FCNs), such as the FCN-VGG architecture (Long et al., 2015). Another noticeable trend in the DNN development was that models become deeper but remained trainable, such as the 19-layer VGG-19 architecture (Simonyan & Zisserman, 2014) or the 152-layer deep DNN achieved using a residual network (ResNet) (He et al., 2016). This advance in computer vision enabled the surpassing of human performance in several specific vision tasks, for example, on the ImageNet Large Scale Visual Recognition Challenge as demonstrated by Langlotz et al. (2019) or for specific agricultural object detection data sets as in Wosner et al. (2021). These two examples demonstrate the potential of DL in person detection and disaster response in general.

2.1 | Semantic segmentation

In this section, we briefly describe the networks considered to select a suitable semantic segmentation model. From base models, also called "feature extractors," more specialized DL models can be constructed to address a specific task. For semantic segmentation, or pixel-wise labeling, the extracted features eventually require

upsampling to generate an output segmentation with the same size as the input image. Current state-of-the-art techniques for benchmark data sets, such as PASCAL VOC,[2] use multiscale features as in MSCI (Lin et al., 2018), PSPNet (Zhao et al., 2017), and DeepLabv3+ (Chen et al., 2018, 2014, 2017a, 2017b). Note that semantic segmentation does not always refer to pixel-wise labeling. For instance, Yuan et al. (2013) requests labelers to segment pictures intuitively, without particular cues, rather than assigning each pixel to a pre-determined category.

Multiscale context intertwining (MSCI) uses an approach in which information between adjacent scales is first exchanged bidirectionally with connections between two long short-term memory chains, referred to as "context intertwining," and then hierarchically combined to calculate a prediction (Lin et al., 2018). In PSPNet, a "pyramid pooling module" has been introduced to gather multi-scale features for use in an ensemble. In this approach, four different pyramid scales are fused through upsampling and concatenating the different feature maps (Zhao et al., 2017). In the DeepLab segmentation model, a similar approach to PSPNet is used. This approach captures image context at multiple scales with "atrous spatial pyramid pooling" (ASPP), which utilizes atrous—also called dilated—convolutions to create multiple effective fields-of-views (Chen et al., 2017a).

We used the DeepLabv3+ approach to segment top-down images from the drone perspective. The main reasons for this decision are that the approach offers (1) sharp boundaries of fine-grained structures, (2) scale invariance to compensate for deviations from the flight height of the training set, and (3) state-of-the-art performance on similar data sets such as the Cityscapes (Cordts et al., 2016) and Skyscapes (Azimi et al., 2019) data sets. For Cityscapes, DeepLabv3+ outperforms ERF-Net (Romera et al., 2017) with 79.6% "mean intersection over union" (mIoU) compared with 72.1% mIoU. For Skyscapes, DeepLabv3+ outperforms U-Net (Ronneberger et al., 2015) with 38.2% mIoU compared with 14.2% mIoU.

For semantic segmentation, we primarily focused on two types of feature detector networks: (1) Xception (Chollet, 2017) and (2) ResNet (He et al., 2016) architectures achieved the best performance on our data sets.

For our application, it was essential to study various feature detectors, as we required to select an (energy-)efficient yet effective compromise for the semantic segmentation step. Relevant considerations and applicable optimizations are summarized in Daghero et al. (2021). Since battery capacity on our drone was limited, it was crucial to select an efficient solution (Chen et al., 2020).

Bhatnagar et al. (2020) demonstrated that semantic segmentation approaches provide higher accuracy for a similarly-sized data set of top-down drone images compared with classical machine learning algorithms. In contrast, Nardari et al. (2018) reported that large semantic segmentation models such as DeepLabv3+ exhibit underwhelming results for small data sets compared with models with fewer parameters. This limitation emphasized the requirement for different feature extractors to facilitate selecting the optimal model capacity for our specific problem (Krajewski et al., 2020). A general method to improve segmentation performance through multi-source ensembles was published by Nigam et al. (2018). However, we avoided using ensemble learning as it would significantly increase computational effort and thus energy usage on our onboard drone system.

2.2 | Object detection

In this section, we briefly introduce recent object detection models and discuss our choice of suitable candidates for person detection. Object detection has a long history in the computer vision field. It initially focused on human detection (Tsukiyama & Shirai, 1985) and was later used to detect human faces (Papageorgiou et al., 1998), cars (Papageorgiou & Poggio, 2000), and so forth.

State-of-the-art computer vision algorithms in the mid-2000s used linear support vector machines (SVMs) (Dalal & Triggs, 2005) or algorithms such as AdaBoost and latent SVM-based classifiers. These classifiers operate on pre-extracted low-level features with methods such as histograms of oriented gradient (HOG), scale invariant feature transformation (SIFT) descriptors, and Haar features (Dollár et al., 2009).

In recent years, these methods have been replaced by significantly more accurate DL approaches, specifically CNNs. OverFeat (Sermanet et al., 2013), and VGG (Simonyan & Zisserman, 2014) can perform object localization using a sliding window approach. This achievement was followed by "two-stage methods" such as R-CNN (Girshick et al., 2014) and R-FCN (Dai et al., 2016), in which a CNN classifies different region proposals.

A very recent development moved toward using simple and faster "one-stage methods," such as single shot multibox detector (SSD) (Fu et al., 2017; Liu et al., 2016) and you only look once (YOLO) (Bochkovskiy et al., 2020; Redmon et al., 2016; Redmon & Farhadi, 2017, 2018). The enhancement primarily results from using a fixed number of default bounding boxes instead of proposal generation.

Object detection models use the same feature extractors discussed previously. For instance, the single-shot detector YOLO uses VGG (Redmon et al., 2016), ResNet (Redmon & Farhadi, 2018), or DarkNet (Redmon & Farhadi, 2017, 2018) as the feature extractor. DarkNet is primarily based on the use of VGG and ResNet. However, it focuses on a low latency to achieve real-time inference. We selected the YOLOv3 model with the DarkNet feature extractor as the first candidate for further comparisons.

RetinaNet (Lin et al., 2017) introduced a novel loss function called "focal loss." It addressed the problem of foreground–background class imbalance while training one-stage object detectors. Specifically, this loss gives less priority to easy examples (inliers) instead of down-weighting complex examples as in robust loss functions, such as the squared error or Huber loss (Hastie et al., 2009). In other words, this means that focal loss puts the training focus on difficult examples. RetinaNet was selected as the second candidate for further comparisons.

[2] http://host.robots.ox.ac.uk:8080/leaderboard/displaylb.php?challengeid=11%26compid=6

However, several other approaches are available. EfficientNet (Tan & Le, 2019) uses a neural architecture search to design a baseline network, which is scalable and has better accuracy and efficiency than previous CNN architectures. EfficientDet (Tan et al., 2020) proposed a network scaling scheme that scales the resolution, depth, and width uniformly for all sub-networks in a coherent manner. As a result, a new class of highly efficient object detectors was developed.

Furthermore, Yu et al. (2020) created a large drone data set with 100 videos that show approximately 2700 vehicles in unconstrained conditions. Based on this data set, they developed the context-aware multi-task siamese network, which improved the robustness for tasks such as single and multi-object tracking. Bargoti & Underwood (2017) used Faster R-CNN to detect fruits on their newly created orchards data set. They provided valuable insights into strategies to generate one's own data set from a field study. In contrast to their Faster R-CNN "two-stage method," we used a simpler and faster "one-stage method" in our study. The decision for a "one-stage method" was supported by the recent publication of Birrell et al. (2020). They demonstrated that YOLOv3 can detect lettuce with a high detection rate of 91% in a field test at realistic conditions. Moreover, we did not consider object detection approaches that use a pipeline of traditional operators, such as Gaussian smoothing, gradient computation, or gradient magnitude thresholding (Leira et al., 2015).

2.3 | Multispectral imaging

In recent years, thermal cameras—in particular those that use uncooled VOx microbolometers—have become more lightweight and affordable. Thus, it has become practical to attach thermal cameras to drones and use them for infrastructure inspection, search and rescue operations, and similar tasks. Multispectral solutions that do not use the thermal band yet exhibit impressive results are available. For instance, Gallego et al. (2019) detected bodies in a maritime setting utilizing the red edge and near-infrared spectrum captured using conventional sensors.

2.3.1 | Multispectral data sets

Several multispectral data sets have been created in the past. For our research, the two most relevant data sets were the KAIST Multispectral Pedestrian Detection Benchmark (Hwang et al., 2015), with 95,000 labeled RGB and thermal images, and the Multispectral Object Detection Data set by the University of Tokyo (Takumi et al., 2017), with 8,000 image tuples of RGB, near-, mid-, and far-infrared images.

Hwang et al. (2015) used a recording setup mounted on a car. They employed sophisticated imaging techniques such as a beam splitter to physically align images with a three-axis camera jig. Takumi et al. (2017) used a less complex setup consisting of three different cameras. Since both data sets focus on the car or pedestrian perspective, we created our own data set from the drone perspective.

The data presented by Takumi et al. (2017) indicated that RGB combined with FIR imaging has exactly the same person detection performance as all four spectral image types combined. Thus, we used only one infrared camera in addition to the RGB camera. Furthermore, this simplified the image alignment as only two cameras with a smaller interocular distance remained. This enabled us to use one single homography matrix, as opposed to programmatically matching the bounding boxes with IDs as proposed in Takumi et al. (2017). In contrast to Hwang et al. (2015), we did not use a beam splitter. Instead, we used the side-by-side mounting similar to that used by Takumi et al. (2017). The main reasons were that the beam splitter setup appeared to be more difficult to assemble, was not as compact in packaging, and, most importantly, would add additional weight to the drone compared with a side-by-side mount. Furthermore, we used a different approach for camera calibration to that described in Hwang et al. (2015). Instead of manufacturing a hole pattern board fabricated from copper, which was used to calibrate the two cameras, we created a simple yet effective calibration pattern using a metallic plate and foam squares that were visible in both RGB and FIR cameras when subjected to heat.

2.3.2 | Multispectral object detection

Two main methods are used to train a multispectral object detection model. The first, more straightforward approach is described in Takumi et al. (2017), in which the authors trained one YOLOv1 model on each of the four input imaging sensors. During inference, those different models were used as an ensemble. Additionally, non-maximum suppression (NMS) was used on all bounding boxes of the four models. The second type of data fusion uses models whose direct input is a multi-channel image. Hwang et al. (2015) modified the aggregated channel features (ACF) pedestrian detector proposed in Dollár et al. (2014) by extending the input channels with (i) a normalized gradient magnitude of the thermal image (TM), (ii) a histogram of oriented gradients of the thermal image (TO), and (iii) the HOG feature of the thermal image.

Other approaches such as Takumi et al. (2017) and Wagner et al. (2016) used DL models to directly process a 4-channel or 6-channel input, respectively. They both observed that models using a multi-channel input—also called early fusion—did not perform better than models using either RGB or thermal images as input. Wagner et al. (2016) speculated that this is due to the limited amount of multi-channel training data in the KAIST data set. However, Wagner et al. (2016) hypothesized that the 6-channel input model could not learn a meaningful feature extraction when the alignment between the channels was poor or when objects were not visually recognizable in one input modality. However, Saha and Mukhopadhyay (2020) criticized that multiple DNNs, used in an ensemble or in late fusion, are not resource-efficient and hence not optimal for feature fusion on resource-constrained mobile devices.

Finally, combinations of the previous two architectures have also been tested. The research of Chen et al. (2018); Konig et al. (2017); Liu et al. (2016); Wagner et al. (2016) can be categorized as early, mid, and late fusion approaches that introduce a fusion layer to fuse the sub-networks at different levels. In our study, we tested the early and late fusion approaches.

Compared with Wagner et al. (2016) and Takumi et al. (2017), our study demonstrated that a resource-friendly 4-channel model can outperform an ensemble approach when the data set is properly aligned. In contrast to Hwang et al. (2015), we used a DL approach instead of classical computer vision algorithms. As an improvement to the research of Takumi et al. (2017), we adapted a more recent and complex object detector for early and late fusion experiments.

Sa et al. (2016) used a "multi-modal" imaging setup consisting of one RGB and one near-infrared sensor to perform object detection for fruits with Faster R-CNN. Similar to their study, we employed two fusion approaches but did not the use the Faster R-CNN model (for reasons described in the previous section). In contrast to our results, Sa et al. (2016) reported that the late fusion approach exhibits superior performance compared with the early fusion method. Additionally, they observed that their RGB model outperformed the infrared model. In contrast, our FIR data clearly outperformed the RGB model. This suggests that in their fruit detection domain, multimodel data fusion was not as effective as that in our domain.

2.4 | AI and sensors on drones

While related studies on sensors on drones have been conducted, none of them satisfies all of our requirements. For example, de Oliveira and Wehrmeister (2018) proposed a pattern recognition system for pedestrian detection from the drone perspective in which they tested three classical machine learning techniques: Haar cascade, SVM classifiers, and one CNN approach (AlexNet). In contrast to our study, the person classifier used only RGB images, whereby thermal images were merely used to generate "regions of interest" proposals for the CNN classifier. Dawdi et al. (2020) presented a system for victim detection using drones. Similar to our study, their system relied on both the visible and FIR spectra for person detection. However, their data fusion step consisted of merely blending the thermal and RGB images linearly using a simple formula with a hand-picked parameter. In contrast to our DL approach, they used traditional computer vision methods such as filtering and edge detection. Person detection was performed with a template matching algorithm based on a hand-crafted human shape template. Such an approach is not robust for detecting humans who appear in various shapes from the drone perspective.

Schedl et al. (2020) described a solution for detecting occluded people from a drone. They combined multiple thermal images from different drone positions and perspectives to remove the occlusions via an algorithm (image registering and integrating). The images from the visible spectrum were only used to estimate the pose of an "RGB–thermal" image pair in relation to other image pairs. Subsequently, person detection occurred in the synthetic, occlusion-removed thermal images with a YOLOv3 detector. In our domain, occlusion is considered the exception and no specific technology is provided. In contrast, our person detector is required to run onboard the drone. Therefore, the technology used by Schedl et al. (2020) was not suitable for our purpose because of (1) slow scan time as many images of the same scene must be captured, (2) long computation time of the pose estimation process on RGB images, and (3) limited applicability as actors for data set recording were instructed to lay on the ground to avoid the "standing person, top-down camera" problem. Our approach overcomes these three limitations.

3 | AI SYSTEM ARCHITECTURE

In this section, we describe the architecture of our onboard AI component. First, in Section 3.1, we explain the overall architecture (Figure 3). Next, in Section 3.2, we describe the Meta Camera, which executes the synchronization and alignment of the three different cameras. Subsequently, in Section 3.3, we explain the improvement of the Geo-Location component (Geraldes et al., 2019) based on a better representation of the surface using terrain data. Finally, in Section 3.4, we describe the interfaces to our two DL components and elaborate on the computation capabilities of the NVIDIA AGX Xavier.

3.1 | System architecture

The AI system architecture is shown in Figure 3. The design of our system enables us to dynamically start and stop tasks as they run in independent threads, whereby each task corresponds to one of the system outputs. The main thread is represented by the Application Manager component that periodically receives telemetry updates from the drone's flight controller and images of the three cameras from the Meta Camera component. When a task is started, a new thread is created, which periodically activates the components involved in the task. The person detection task first activates the Person Detector component, which detects a person in the 45° images and outputs the result to the Geo-Location component, which computes the corresponding geographic position. The final result is the number and locations of the detected people. The Application Manager, Task Manager, and Meta Camera components were developed in C++ using OpenCV and Boost. The other components—Semantic Segmentation, Landing Spot Detector, Road Blockage Detector, Person Detector, Geo-Location—were implemented in Python using TensorFlow and PyTorch. The C++ application embeds Python and invokes each component's functions as required.[3]

[3] https://docs.python.org/3/extending/embedding.html

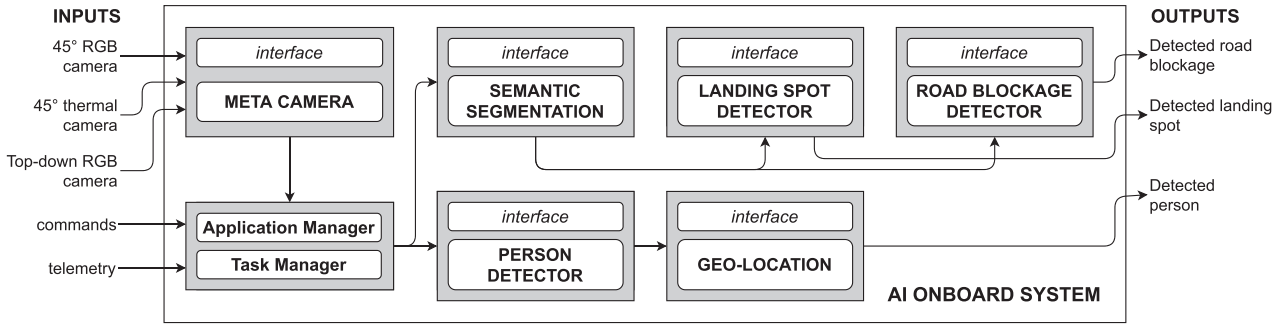

FIGURE 3 Onboard application architecture of the system running on the NVIDIA AGX Xavier. The outputs are sent to the drone's flight controller.

To facilitate the use of Python in the C++ application, we implemented all Python components with the same version (Python 3.8).

The tasks are managed via input commands. Thus, the application can run multiple tasks simultaneously. This provides high flexibility to complete the challenge, for example, by performing road blockage detection and person detection simultaneously.

The AI components are deployed on an NVIDIA Jetson AGX Xavier system-on-a-chip (SoC), which is mounted on top of the drone next to the GPS and communication antennas. The major benefit is its low power consumption of 30 W and its high computational performance of 32 TOPs. The AGX Xavier has 16 GB of shared memory which enables us to execute inference of two DL models simultaneously and access multiple video input streams simultaneously. The drone and AGX Xavier are powered by one 364.8 Wh battery, which enables the drone to fly for approximately 25 min. The consumption of 30 W of the AGX Xavier represents only 3.4% of the total power consumption.

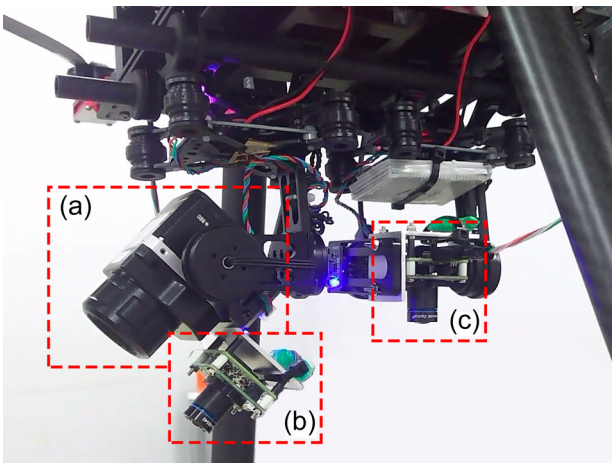

FIGURE 4 FLIR Vue Pro 640 thermal camera (a) mounted on a gimbal system under the drone. Two e-con Systems See3CAM were installed: one (b) was rigidly attached to the FLIR camera with a 45° forward tilted angle and another (c) faced 90° downwards for segmentation task. (a) and (b) were used for human detection. This is a picture without the lens hoods installed.

3.2 | Meta camera component

Our system uses three cameras that are accessed by the Meta Camera component: two RGB cameras and one thermal camera. While one RGB camera, pointed downwards, is used as the input for the semantic segmentation component, the Person Detector component uses both RGB and thermal cameras, which are pointed at 45°. The cameras are assembled on a gimbal, as seen in Figure 4, which keeps them stable, independent of the movement of the drone. The Meta Camera component is responsible for the inputs of the three cameras, synchronizing them with the drone telemetry, and aligning the RGB and thermal images. The alignment algorithm is implemented with a homography transformation, as described in Appendix B. The homography was estimated once for a particular physical camera configuration using OpenCV during calibration. Therefore, we manually picked four pairs of points in a pair of images. We obtained a reprojection error of approximately 5 pixels with this simple alignment method for images with a size of 3840 × 2160. The homography does not mathematically describe the relation between the two images, but because the distance between the cameras is small (approximately 5 cm), whereas the distance to the objects of interest is large (over 20 m), this is a reasonable approximation.

3.3 | Geo-location component

Given a pixel position in an image, for example, one corresponding to a person detected by the DL models, we estimate its geographic position (latitude and longitude in the World Geodetic System 1984 [WGS 84]). Thus, we use the drone's GPS position reading, height, orientation, and the camera's orientation and lens characteristics. Our method is based on that of Johnston (2006) and considers the camera intrinsic parameters, as described by Geraldes et al. (2019). We also consider the elevation of the terrain, using the approach by Salamí et al. (2013). This approach is an improvement over the original method, where the terrain was assumed to be flat and mountains and hills were ignored.

Salamí et al. (2013) permitted the camera to have a non-zero roll angle, while our approach assumes the roll to be zero. In our particular application, the pitch is constant (45°), but our method supports other values. The roll is assumed to be zero because of the use of a gimbal that physically guarantees it. Salamí et al. (2013) did not consider lens distortion, but since this is a simple enhancement, we included it in our implementation. To address terrain elevation, Salamí et al. (2013) used an iterative method that converges toward the target.

The detailed algorithm and formulas of the geo-location method, and the consideration for elevation, are explained in Appendix C. The Geo-Location component was implemented in Python using OpenCV and runs in 1.24 ms on the AGX Xavier.

3.4 | AI components

Our AI onboard system consists of two core DL models. The semantic segmentation DL model is used to detect road blockages and landing spots and was implemented in Python with TensorFlow. The object (person) detection DL model is used to detect people and report their location and was implemented in Python using PyTorch.

Note that since road blockage detection is a rather openly formulated task, we built it on top of the eight-class segmentation mapping of our semantic segmentation model. The benefit of this method is that we do not require to know in detail the appearance of road blockages in the challenge. Instead, we can formulate an algorithm that assumes that paved ground is safe for vehicles to drive on, and anything else (the remaining seven semantic classes) is a potential obstacle that must be identified. For the landing spot detector, we require the segmentation output to identify spots on which a drone cannot land, such as on plants or in water.

For the person detector, the outputted bounding boxes are directly used to compute the geographic positions of the detected people. This computation is performed in the Geo-Location component (Section 3.3), which considers the bottom center pixel location of each detected bounding box as input and computes the corresponding geographic position. The geographic position is then sent to the human operator in the disaster operation control room.

4 | SEMANTIC SEGMENTATION MODEL

In this section, we present the model used for semantic segmentation. Well-known methods from the literature were applied to a different type of data set, that is, drone views, rather than "pedestrian" views. Our results are compared with a previous approach (Laurmaa, 2016) that operates on a similar data set. In Section 4.1, we recap the metrics for semantic segmentation. Section 4.2 presents the used model with its encoder–decoder structure, which promises to improve on previous studies owing to the use of multi-scale contextual information. Here, we compare several state-of-the-art feature extractors. The aim is to identify the one with which the model achieves the highest mIoU. Our results, in Section 4.3, show that our suggested model outperforms previous models. We also provide visual evidence of its performance. Finally, Section 4.4 describes the use of the semantic segmentation model for landing spot and road blockage detection.

4.1 | Semantic segmentation metrics

Semantic segmentation describes the problem of mapping each pixel $\boldsymbol{x}^{(m,n)}$ of a 2D input image $\boldsymbol{X}$ to its predicted class $y^{(m,n)} \in \{1, 2, \ldots, c\}$, where c is the number of predicted classes. This is achieved by a feature extractor that primarily consists of 2D convolution operations and outputs feature maps, given the size of the downsampled input image. These extracted feature maps are then used by the upscale sub-model to compute a logit tensor, containing logit vectors $\boldsymbol{z}^{(m,n)}$ for each pixel of the input image $\boldsymbol{x}^{(m,n)}$. The value of element $\boldsymbol{z}^{(m,n)}(i)$ equals the probability estimated by the model that pixel $\boldsymbol{x}^{(m,n)}$ belongs to class i. Finally, the output label is computed as $y^{(m,n)} = \arg\max_i \boldsymbol{z}^{(m,n)}(i)$.

However, it should be noted that in downstream tasks of the segmentation model, the logit tensor can be more useful as it contains more detailed information than the output labels. In our use case, it is beneficial to additionally use the logit matrix of the classes "paved ground" and "vehicles" to achieve a better understanding of road boundaries and road blockages. For landing spot detection, the logit matrix of "outdoor structures" and "water" may be of high relevance as we seek to avoid landing on those areas.

We use the (pixel-wise) accuracy as one of the key metrics to evaluate and compare segmentation performance. It expresses the percentage of input pixels $\boldsymbol{x}^{(m,n)}$ in the data set predicted to have the same label $y^{(m,n)}$ as annotated in the ground truth $g^{(m,n)}$.

However, owing to a class imbalance in our data set toward "non-paved ground" and "plants," the pixel-wise accuracy metric is not suitable to draw conclusions for underrepresented classes such as "train tracks" or "water." To solve these problems, we the mIoU metric. The definition of mIoU is shown in the following equation:

$$\text{mIoU} = \frac{1}{c}\sum_{i=1}^{c}\frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i + \text{FP}_i}. \quad (1)$$

Mathematically expressed, it is the average of per-class Jaccard indices. In Equation (1), TP_i denotes true positives in the ground truth class i. FN and FP denote false negatives and false positives, respectively. The Jaccard indices for each class (IoU) are later shown for each individual class, data set, and model to provide a precise insight into how thoroughly the models learned a specific class.

4.2 | DeepLabv3+ encoder–decoder model using different feature extractors

In this study, the architecture proposed by DeepLabv3+ (Chen et al., 2018) was used. DeepLab adds an encoder–decoder structure to state-of-the-art feature extractors such as ResNet or Xception.

The motivation to use DeepLabv3+ is threefold. First, it achieved excellent performance on many segmentation data sets as of October 2019. Second, the codebase was well maintained and documented, which facilitated adaptions to our problem. Specifically, the implementation of the model and its training process was performed using the DL framework TensorFlow and the on top of high-level library TensorFlow-Slim (Abadi et al., 2016; Chen et al., 2018; Sergio & Nathan, 2016). Finally, the authors shared pre-trained weights for five different feature extractors, which made DeepLab highly configurable in terms of segmentation performance and runtime.

The DeepLab encoder is constructed by modifying one of the previously mentioned feature extractors, which are also called the "backbone." The fully-connected layer for final image classification is removed along with the pooling layer to use only the final feature maps.

Four different feature extractor model types from the ResNet and Xception model families were compared. The different backbone models can be used interchangeably with the same DeepLab encoder–decoder structure. All models were trained and evaluated on the Okutama Drone and Swiss Drone segmentation data sets. ResNet (He et al., 2016) enables the training of deeper CNNs. Commonly used variants of this architecture are ResNet-{50, 101, 152}. We used the two smaller variants to maintain a feasible computing budget for our onboard inference. Xception (Chollet, 2017) is an improvement of the Inception V3 architecture (Szegedy et al., 2016).

4.3 | Results for semantic segmentation models

Segmentation data sets are freely available for images of common objects, such as Everingham et al. (2010); Lin et al. (2014); Russakovsky et al. (2015), or for images captured from car perspectives for autonomous driving (Cordts et al., 2016). To account for the drone perspective, we previously prepared two aerial segmentation data sets, which can be used in several tasks related to environment recognition by UAVs.

The *Swiss Drone* data set was recorded around Cheseaux-sur-Lausanne in Switzerland using a senseFly eBee Classic in 2014 (SenseFly, 2020). The 100 images were captured from a top-down perspective at a flight height of approximately 80 m above the ground at a resolution of 4608 × 3456 pixels. The *Okutama Drone* data set was recorded and annotated by NII (Laurmaa, 2016) in 2016 using a DJI Phantom 4 at a resolution of 3840 × 2160 pixels. The 91 images were captured over Okutama, west of Tokyo, Japan, from a drone at a flight height of approximately 90 m above the ground. Here, the flight height may have varied more as Okutama is located in a narrow valley with uneven ground. Both data sets were labeled with the same nine classes, including "buildings," "paved ground," and "water" (Figure 2).

The DeepLabv3+ models were trained on the Okutama Drone and Swiss Drone data sets with heavy use of data augmentation. During the training process, we randomly scaled the input images by a factor between 0.25 and 1.5, used random crops, randomly mirrored the image at the horizontal and vertical axis, used 90° lossless rotation, and used bilinear rotation between 0° and 360° as top-down aerial footage is rotation invariant. As shown in (Liu et al., 2021), this approach aids in mitigating the overfitting problem as it increases the training data set.

Figure 5 shows that our segmentation data sets are not well balanced. To account for that imbalance, we modified the softmax cross-entropy loss by multiplying the losses of each class by the inverse of its class occurrence. This provided a higher loss to classes that are underrepresented in the data sets. Our experiments demonstrated that this loss weighting resulted in an mIoU increase of approximately 3 percentage points, whereas pixel accuracy did not improve.

4.3.1 | Comparison to previous research

First, our new DeepLabv3+ model with ResNet-101 backbone was compared with the metrics stated in Laurmaa (2016) to verify if the use of the encoder–decoder structure improves on previous research. Next, we compared four feature extractors on the Okutama and Swiss Drone validation data sets as a hyperparameter selection step.

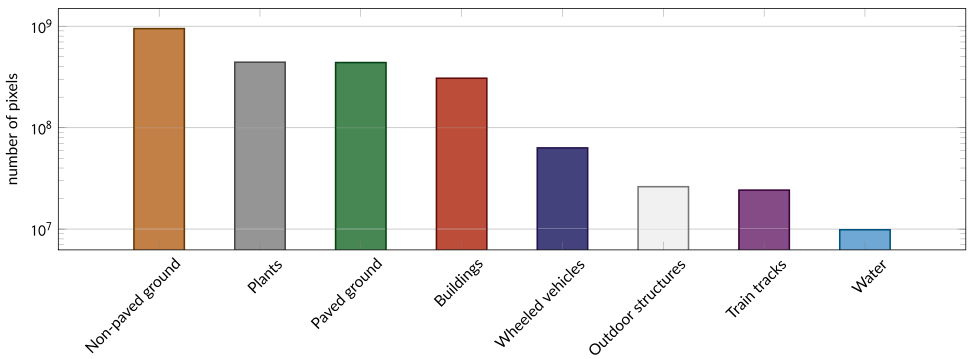

FIGURE 5 The number of labeled pixels per class in the Okutama and Swiss segmentation data sets. The total number of pixels in both data sets is $2.35 \cdot 10^9$.

TABLE 1 Comparing our new model with encoder-decoder structure to three models (1)–(3) in Laurmaa (2016) on the Swiss 10 class data set

Model	Acc	mIoU
(1) FCN-ResNet-152-skip	95.4%	67.3%
(2) FCN-ResNet-152-skip augmented	96.0%	68.2%
(3) 3 model ensemble	**96.2%**	69.7%
DeepLabv3+-ResNet-101	96.0%	**70.6%**

Note: Bold values indicates the best value on a given column.

We begin with the comparison of DeepLabv3+-ResNet-101 to previous research (Table 1). For the sake of comparison, we trained a ResNet-101 model for 600,000 iterations on the ten-class[4] Swiss data set and achieved 70.6% mIoU. This result indicated that even the single model with DeepLab encoder–decoder structure outperformed the previous ensemble model (Laurmaa, 2016) by 0.9 percentage points. This result suggested that the performance improvement can be attributed to the DeepLabv3+ encoder–decoder architecture, as it outperformed the ensemble model of Laurmaa (2016) despite its less effective feature extractor (ResNet-101 vs. ResNet-152). However, the pixel-wise accuracy of our model was 0.2 percentage points lower than the ensemble model and achieved exactly the same accuracy as the data augmented model (Table 1). This may be because the class weighted loss function improves mIoU scores while slightly lowering accuracy scores.

Table 1 shows that Laurmaa's (Laurmaa, 2016) best model (FCN-ResNet-152-skip) achieved 68.2% and 67.3% mIoU with and without data augmentation, respectively. Random flip and random crop were used as data augmentation techniques. The overall best performance was achieved using an ensemble model consisting of three models, each with different architectures and training processes. This ensemble achieved 69.7% mIoU on the Swiss data set with all 10 classes predicted (including the classes "background" and "person").

4.3.2 | Comparison of different feature extractors

Finally, we compare different feature extractors on the Okutama and Swiss Drone data sets. Specifically, we show the results of comparing DeepLabv3+ models with different backbones. We trained and evaluated all possible configurations of models on each of the threefolds. This resulted in three training runs on every folds training set, with one single training run using 250,000 iterations. This was conducted for each of the four models. Finally, we evaluated the models on the respective fold's test set and averaged all metrics over the threefolds per model. The test results are shown in Table 2.

[4]In the comparison in Table 1, all four models were configured to additionally predict the classes "person" and "background," which lowered the mIoU value compared with models shown in Table 2.

During previous research on that data by (Laurmaa, 2016), we encountered some overlapping between training, validation, and test data splits. Furthermore, owing to the effort of manually labeling all images, we aimed to avoid letting a large portion of the data set remaining unused for training as it would be used for validation and testing. Therefore, we decided to use threefold cross-validation in combination with a data set split into 12 geographic regions, whereby we controlled and minimized the inter-region overlap.

Thus, the data set was split as follows: (1) Each original resolution image was subdivided into four equally sized, smaller resolution images in a 2×2 pattern. The resulting images had sizes of 2304×1728 and 1920×1080 pixels for data recorded in Switzerland and Okutama, respectively. (2) All 764 images from the previous step were separated into 12 geographic regions (Figure 6) and the overlap was minimized by splitting the regions along the shorter sides, leading to shorter boundaries. We additionally discarded some images at the boundaries to maintain the overlap between any two regions at or below 5%. The overlap was measured as the overlapping area when projecting the field of view of each image into a map by using our geo-location method. (3) Finally, we divided the 12 geographic regions into training, validation, and test splits for each of the threefolds used in the threefold cross-validation procedure. The results of this random process are shown in Table 3.

We determined ResNet-101 to be best suited for the tasks in an actual disaster as it has a smaller memory footprint and latency than both Xception models tested while exhibiting optimal segmentation performance. While ResNet-50 is the most efficient model, it clearly lacks segmentation performance compared with ResNet-101 (ResNet-50 was down 12 percentage points in acc and 22 percentage points in mIoU). When evaluating validation metrics and training losses during the training process, we observed that the two models that performed significantly worse, namely ResNet-50 and Xception–71, under-fitted as the validation metrics were still improving at the 250,000th step.

We calculated that, with the drone flying at an altitude of 35 m, the top-down field-of-view would be approximately a rectangle of 71×52 m. If the drone flew at a speed of 20 ms^{-1}, it would require 2.64 s to travel 52 m and capture a completely new scene. Therefore, the semantic segmentation latency is limited by this time interval, that is, it must be lower such that some overlap occurs between consecutive frames, no area is missed out, and there is sufficient processing time to also conduct person detection. Regarding memory consumption, both the semantic segmentation and person detection models must fit in the NVIDIA AGX Xavier's RAM (16 GB).

Figure 7 provides qualitative visual results of our model (ResNet-50 as running on the Xavier) on examples from the training and validation data sets.

4.4 | Semantic segmentation for landing spot and road blockage detection

In this section, we will describe the use of semantic segmentation with RGB images for landing spot and road blockage detection.

TABLE 2 Test results from the threefold cross-validation on the Okutama Drone and Swiss Drone data sets

Model	Acc Swiss	Acc Okutama	mIoU Swiss	mIoU Okutama	Acc	mIoU	mem (GB)	lat (s)
DeepLabv3+-ResNet-101	**91.73**	89.83	**63.86**	67.89	**90.78**	**65.88**	3.71	2.023
DeepLabv3+-ResNet-50	80.56	76.74	41.74	45.55	78.65	43.65	**3.02**	**1.669**
DeepLabv3+-Xception-71	77.76	70.86	38.53	37.09	74.31	37.81	4.60	2.705
DeepLabv3+-Xception-65	91.29	**90.15**	60.20	**68.47**	90.72	64.34	4.54	2.397

Note: The test input resolution of the models was the resolution of the 2 × 2 subdivided data set. Accuracy (Acc) and mean intersection over union (mIoU) at test time are separately calculated on the parts from the Okutama Drone and Swiss Drone test splits and averaged over all threefolds. Acc and mIoU is the arithmetic mean between the two data set values. Memory usage (mem) in gigabytes on FHD input images. Latency (lat) on 4K images in seconds, as run on the NVIDIA AGX Xavier.

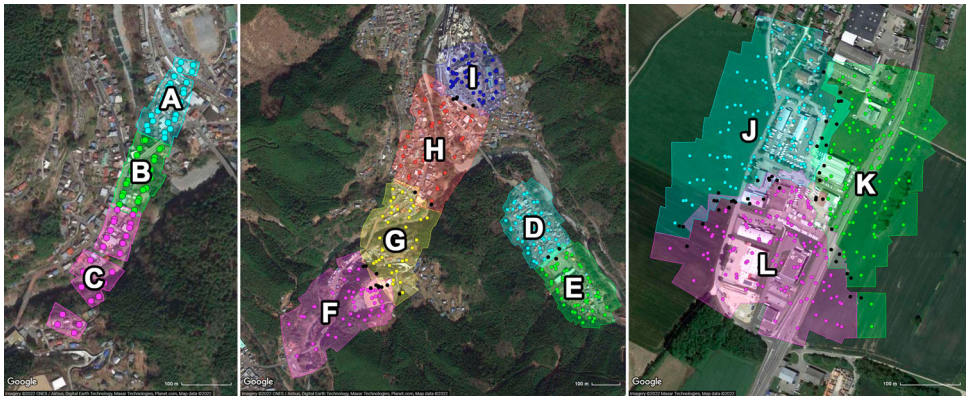

FIGURE 6 Splitting of Okutama and Swiss data sets into 12 geographic regions labeled A to L. The two leftmost images visualize the two flights in Okutama. The rightmost image shows the flight in Switzerland. Each colored dot represents the center of an image from the 2 × 2 subdivided data set. The black dots represent images that were excluded to reduce the region overlap.

TABLE 3 Distribution of our 12 geographic regions into train, validation and test set for the threefolds

Splits	Training	Validation	Test
Fold 1	D, H, F, K, E, C, I	L	G, A, B, J
Fold 2	E, C, I, L, G, A, B	J	D, H, F, K
Fold 3	G, A, B, J, D, H, F	K	E, C, I, L

Note: Single letters correspond to the letters A to L in Figure 6.

Our system solely focused on the detection of a suitable landing spot, whereas other systems, such as Pijnacker et al. (2018), studied the control system of a flying robot to achieve a fast landing maneuver.

4.4.1 | Landing spot detector

The Landing Spot Detector uses the output of the semantic segmentation to determine a suitable place for the drone to land, away from potential hazards and obstacles. Note that this is not a precision landing system since it does not continuously steer the drone toward a well-defined point on the ground. Instead, it suggests a landing spot from one snapshot of semantic segmentation and estimates its latitude and longitude coordinates.

This method uses the drone's telemetry and an initial target point as input, which can be, for example, the point directly below the drone. The procedure is described in Algorithm 1 and can be summarized as follows.

- Semantic segmentation is used to create a binary image of suitable landing areas, "paved ground" and "non-paved ground," painted in white. All other classes are considered "non-landable" and painted in black.
- The white "landable" area in the binary image is shrunk by 5 m to leave a safety buffer of 5 m between landable and non-landable areas. The shrinking is implemented as an image "erosion," i.e., any point in the resulting smaller areas will be at least 5 m away from the borders with the non-landable zones. We use OpenCV's implementation of erosion.
- We locate the white pixel closest to the given target point.
- Finally, we convert the observed point with the Geo-Location component to obtain the latitude and longitude of the landing spot.

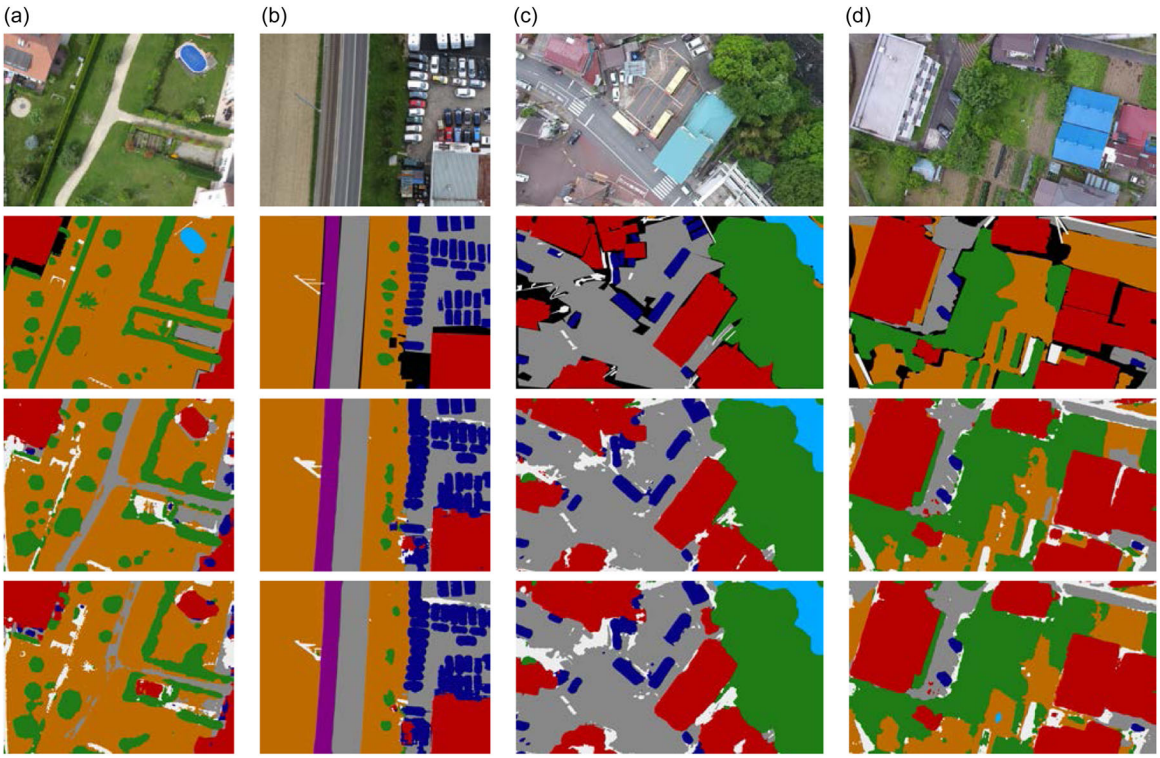

FIGURE 7 Visualization of a subset of test results from the Okutama and Swiss segmentation data sets. Swiss images by courtesy of senseFly. For color coding, please refer to Figure 2. From top to bottom: (first row) input image, (second row) ground-truth, (third row) ResNet-101 segmentation as run on the Xavier, and (fourth row) Xception-65 segmentation as run on the Xavier. (a) Swiss Fold 1, (b) Swiss Fold 2, (c) Okutama Fold 2, and (d) Okutama Fold 3.

Algorithm 1 Landing Spot Detection

Require: segmentation, drone telemetry

image $landable \leftarrow$ all areas segmented PAVED GROUND or NON-PAVED GROUND colored 1, rest colored 0
$buffer \leftarrow$ metersToPixels(5) ▷ Convert 5 m to pixels in the current image, depending on drone telemetry
$landable \leftarrow$ erode($landable$, kernel($buffer, buffer$)) ▷ Using OpenCV erode, with a square kernel for faster processing
$target \leftarrow landable.center$ ▷ Aim to land close to the center of the image, directly below the drone
$landablePoints \leftarrow cv2.findNonZero(landable)$ ▷ List of landable pixels on the image
$distances \leftarrow (landablePoints - target)^2$ ▷ Compute all squared distances from all points to target; avoid sqrt for performance
$nearestIndex \leftarrow np.argmin(distances)$ ▷ Determine closest landable point to our target
$nearestPoint \leftarrow landablePoints[nearestIndex]$
return getGeolocation($nearestPoint$) ▷ Get latitude and longitude of observed point

4.4.2 | Road blockage detector

In a disaster, roads may be blocked by mud, trees, or other debris, owing to landslides. From the perspective of the drone, it may not be possible to tell if there was a road in a certain place. To address this problem, we used OpenStreetMap (OSM)[5] as ground truth data, and we cross-referenced it with the result of the semantic segmentation. If a segment of a road exists in OSM but is not detected in the segmentation, we report the corresponding point as a road blockage. The method is implemented as a mask-and-difference operation. The pseudo-code is shown in Algorithm 2.

4.4.3 | Runtimes of landing spot and road blockage detectors

We evaluated the runtimes of both the Landing Spot and Road Blockage detectors on the target hardware (NVIDIA AGX Xavier). Table 4 presents the averaged results of running both components on

[5]https://www.openstreetmap.org

Algorithm 2 Road Blockage Detection

Require: segmentation, drone telemetry, map data from geographic information system (GIS)

Align image with map data using Geo-Location
image $paved \leftarrow$ image of all "PAVED ROAD" areas in segmentation as 1, rest as 0
image $roadmask \leftarrow$ blank image (all 0)
set of roads $R \leftarrow$ get visible roads in image from GIS
for all $r \in R$ **do** ▷ Create mask where roads from GIS are filled as 1, rest as 0
    $w \leftarrow$ compute width of road from meters to pixels in image
    drawLine(r, image: $roadmask$, width: w, color: 1) ▷ Using Reverse Geo-Location
end for
$diff \leftarrow roadmask - paved$ ▷ Find areas that are indicated as roads in GIS but absent in segmentation; clamped to $[0, 1]$
$blobs \leftarrow$ getBlobs($diff$) ▷ Using OpenCV findContours
for all $b \in blobs$ **do**
    $P_{geo} \leftarrow$ getGeolocation(getCentroid(b))
    add P_{geo} to $blockages$
end for
return $blockages$

TABLE 4 Performance metrics for landing spot and road blockage detectors, measured on 91 images of size 3840 × 2160 and averaged

Component	Runtime (ms)
Landing Spot Detector	164.6
Road Blockage Detector	21.4

Note: Results exclude the runtime of the segmentation model.

the 91 images of the Okutama data set, which have a size of 3840 × 2160.

5 | CREATION OF MULTISPECTRAL AERIAL PERSON DETECTION DATA SET

In this section, we explain the creation of the National Institute of Informatics and Chiba University (NII-CU) Multispectral Aerial Person Detection data set. First, in Section 5.1, we describe our drone setup to record the data set and justify our selection of cameras. Second, in Section 5.2, we explain the recording and labeling process of the data set and provide some statistics about the data set. Finally, in Section 5.3, we compare our data set to two similar data sets in the field, namely, the KAIST and TODAI multispectral person detection data sets.

Similar to the segmentation task, many multispectral data sets from a car's perspective are available (Hwang et al., 2015; Takumi et al., 2017). The difference to our application is that in these automotive data sets, humans appear in their full profile. However, we use a 45° angle view to detect people. This angle originated as a compromise between top-down imaging, where people are only poorly visible, and a similar frontal angle, such as from a car's perspective. Flight height is important as we aim to record humans above a specific minimum pixel height, yet we are also limited in resolution with the thermal camera. For example, Hwang et al. (2015) stated that pedestrians that appear smaller than 45 pixels are considered far away; thus, person detection is ineffective. They also considered people larger than 115 pixels as near (they used a 640 × 512 resolution in RGB and FIR spaces).

However, in our scenario, people can never appear nearer than the flight height h with a distance at the image center of $\sqrt{2}h$ in the case of 45° angled RGB and thermal cameras. Interestingly, with our four times higher resolution in the RGB images, people had a similar size in terms of pixel height compared with Hwang et al. (2015). This applies to RGB images only as the data set was labeled in the RGB space (Figure 8).

5.1 | Drone setup for data collection

Since existing pedestrian detection and other top-down aerial view data sets cannot be used for this task, we created our own data set, which we refer to as the NII-CU Multispectral Aerial Person Detection data set (see Table 5 for a comparison).

We first evaluated which thermal camera to use. This step was essential for selecting equipment suitable for person detection because different thermal cameras can detect different wavelengths. Jones (1998) stated that human skin can be approximated as a black body radiator; thus, the peak emission wavelength of human skin is only dependent on its temperature. With measurements conducted in (Humamatsu Photonics, 2011), where clothing and season were considered for temperature measurements, they observed that human surface temperature ranges from 20°C to 32°C. Through black body approximation (Jones, 1998) and using Wien's law, this yields a peak emission wavelength (λ_{max}) between 9.9 µm (corresponding to 20°C), 9.5 µm (corresponding to 32°C), and 9.3 µm (corresponding to a body temperature of 37°C). The latter number was also stated in St-Laurent et al. (2007).

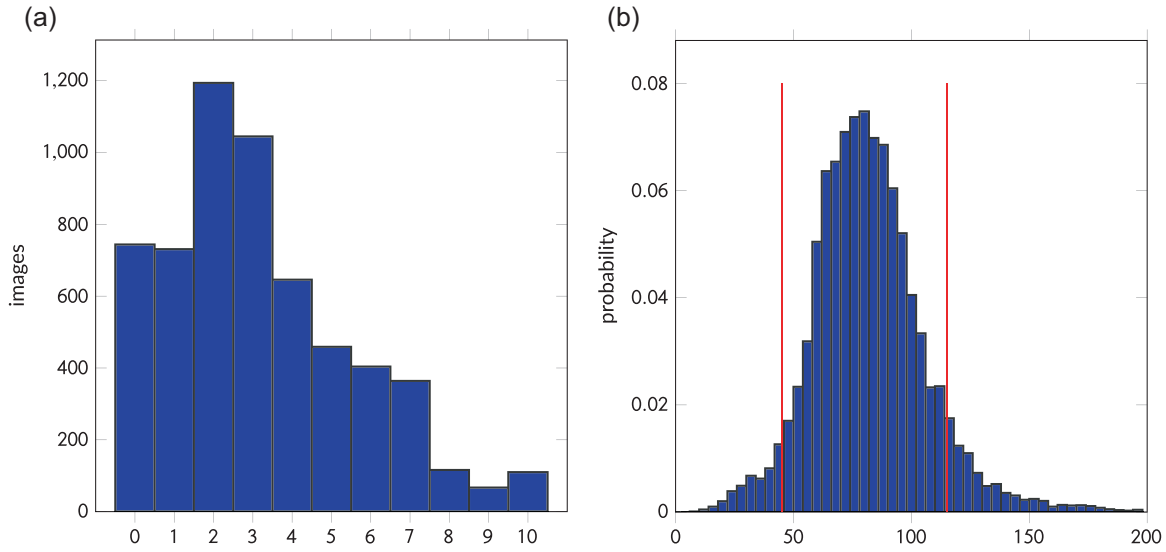

FIGURE 8 (a) Most frames in the data set contain two persons with 744 images containing no person. (b) Distribution of person bounding boxes height as they are labeled in the data set in 4K resolution. We marked 45 and 115 pixels as Hwang et al. Hwang et al. (2015) referred to as near and far. (a) Number of persons per frame and (b) distribution of person height.

Thus, all numbers indicate that our target spectrum to detect humans is the FIR spectrum. Considering this, we used a FLIR Vue Pro thermal camera with a spectral detection band of 7.5–13.5 µm.

To record the data set, we used a DJI Matrice 100 with a standard Zenmuse X3 RGB camera mounted on a gimbal. The FLIR thermal camera was rigidly mounted close to the onboard RGB camera. We ensured that both cameras pointed in the same direction since this was crucial for later alignment. An example of raw images and aligned overlay is shown in Figure 9. The RGB video was recorded at a resolution of 3840 × 2160, while the thermal camera recorded at a resolution of 640 × 512. Both cameras were recording at 30 Hz, and the videos were manually synchronized after recording. The full camera specs are shown in Table 6. Note that a different RGB camera was used for the proposed onboard system because it is not possible to access the live feed from the Zenmuse X3 camera in real-time through the onboard computer (Appendix B). However, we used images captured with the Zenmuse X3 camera for the RGB results reported in this paper.

The drone was mostly flying at a height between 20 and 50 m above the ground. Some of the labeled frames are shown with the ground truth bounding box overlays in Figure 10.

5.2 | Multispectral data collection

We conducted three flights on a baseball field in Chiba and recorded a total of 34 min of video. We extracted 5880 images of each RGB and thermal video and aligned them using the algorithm described for the Meta Camera component (Section 3.2). Sample frames are shown in Figure 10.

In these frames, we annotated 18,736 instances of people as ground truth, which consisted of bounding box coordinates and labels indicating various conditions of alignment and visibility. For example, if a person was perfectly aligned in the RGB and FIR images, we annotated that person with one single, tight bounding box with the label "both." If the person did not align perfectly in the RGB and FIR image, we used two bounding boxes labeled with "RGB" and "thermal," respectively. Thus, we filtered out poor alignments in advance during labeling and saved labeling effort by labeling RGB and FIR simultaneously where possible.

Compared with Takumi et al. (2017), the alignment of images before labeling achieved a visually superior alignment compared with annotating the same people with matching IDs in the different modalities of an image and automatically computing homography matrices.

Our data set was split into 4980 training and 900 test frames, whereby we ensured that the two sets had no overlap; thus, the test data was unseen during training. The training set focused on the baseball field and dense woods, whereas the test set displayed a different area of the woods and a parking lot, with cars and a house in the background.

We prepared a subset of the data that contained only images in which humans in RGB and FIR were perfectly aligned (labeled "both"). Additionally, we excluded images with any misalignment or in which a person was visible in one image but not in the other. We also cropped these images to include only the area in which the RGB and FIR fields of view overlapped (Figure 11). This subset contained 3138 4-channel images (2653 for training, 485 for testing), which were used to train a 4-channel model.

The data set contains humans in different clothing walking on a field, hiding behind trees, and standing in between bushes. The actors (students and researchers from Chiba University and NII) were instructed to behave as either a rescuer (e.g., wearing a helmet, walking) or as a missing person (e.g., waving arms, sitting, or laying on

TABLE 5 Comparison of multispectral data sets ($k = 10^3$) from The Korea Advanced Institute of Science and Technology (KAIST), The University of Tokyo (TODAI), and our NII data set

	Training		Testing		Properties							
	# persons	# images	# persons	# images	# total frames	drone perspective	RGB: 400–800 nm	NIR: 0.8–3 µm	MIR: 3–6 µm	FIR: 6–15 µm	Publication
KAIST Hwang et al. (2015)	41.5 k	50.2 k	44.7 k	45.1 k	95.3 k		✓			✓	'15
TODAI Takumi et al. (2017)	–	1.6 k	–	1.4 k	7.5 k		✓	✓	✓	✓	'17
NII-CU (Ours)	16.7 k	5.0 k	2.0 k	0.9 k	5.9 k	✓	✓			✓	'22

Note: # images refers to the number of image pairs (in case of TODAI, it refers to the number of image 4-tuples).

the ground). We also included scenes that are challenging to correctly classify for the thermal detection model, such as placing warm, worn jackets or helmets on the ground, a cold parked car, and a warm running car in the background. The ground on the location consisted of grass, dirt, woods, and asphalt that provided various backgrounds.

5.3 | Comparison with other multispectral data sets

Our data set is similar to the data set of Takumi et al. (2017) in that it uses separate cameras for RGB and thermal data collection. In contrast to Takumi et al. (2017), in our application, the cameras are mounted on a flying UAV; thus, they are always at a relatively high distance from the objects of interest. The people are always more than 20 m away from the cameras. In contrast, the distance between the principal points of the cameras is only a few centimeters in our setup. We also only align two cameras instead of four. This means that the homography-based alignment is effective regardless of the distance to the objects. This condition enables successful training of a 4-channel (RGB plus thermal) model. Table 5 compares the KAIST (Hwang et al., 2015) and TODAI (Takumi et al., 2017) data sets to ours. We indicate the different image types provided and the wavelengths of the used cameras. As the TODAI data set (Takumi et al., 2017) provided four different image types, we used it to verify our camera selection.

The KAIST data set (Hwang et al., 2015) features significantly more images in different settings but is limited to the perspective of a car similar to the TODAI data set. We adapted some labels from KAIST, such as occlusion tags that allow for more detailed experiments in the future. Our data set is the only one containing well-aligned aerial RGB and thermal images with a 45° viewpoint. In addition, our approach contains occlusion labels and people in different environments and backgrounds being useful for person detection in a real scenario. More importantly, our RGB imaging quality is higher than in the other data sets to detect people even from far distances.

6 | PERSON DETECTION MODEL

In this section, we describe a novel DL person detection model. Section 6.1 describes our metrics for object detection. Section 6.2 reports on the performance of YOLOv3 and RetinaNet, as single-spectral models, using RGB and thermal (FIR) images only. Section 6.3 describes the architecture of multispectral DL models based on the YOLOv3 architecture. Finally, in Section 6.4, we compare the performance of an ensemble model and several 4-channel models.

6.1 | Object detection metrics

Object detection describes the extension of the image classification and object localization problem to predicting multiple objects. Formally defined, for every input image $\boldsymbol{X}$, a set of M predicted

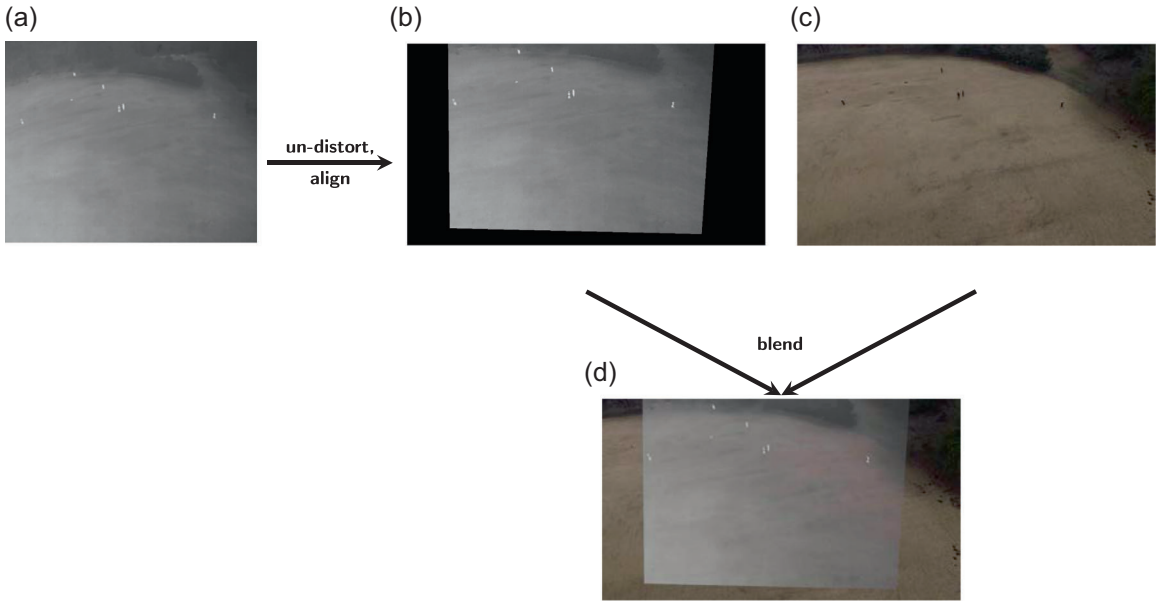

FIGURE 9 Un-distortion and alignment process of an RGB, thermal, image pair captured by the drone during flight. Image (b) is generated from image (a) by using the un-distortion and alignment algorithm described in Appendix B, Algorithm 3. Image (d) is generated as a blended overlay of the two already aligned images (b) and (c) and visualizes the correct alignment. (a) Raw frame from thermal camera, (b) undistorted and aligned thermal image, (c) raw frame from RGB camera, and (d) Blended overlay of images (b) and (c).

TABLE 6 Specifications of the cameras used for data collection and onboard inference

Camera	Thermal	RGB (onboard)	RGB (data collection)
Model	FLIR Vue Pro 6409 mm	e-con Systems See3CAM CU135	DJI Zenmuse X3
Spectral band	7.5–13.5 µm	0.4–0.8 µm	0.4–0.8 µm
Sensor size	10.88 8.704 mm	4.629 3.432 mm (1/3.2”)	8.835 6.626 mm (1/2.3”)
Pixel size	17 µm	1.1 µm	2.1 µm
Resolution	640 × 512	2880 × 2160	3840 × 2160
Frame rate	30 Hz	20 Hz	30 Hz
Lens	Built-in 9 mm FL	EdmundOptics 3 mm FL f/2.5	Built-in 20 mm FL (35 mm eq.) f/2.8
Field of view	69° × 56°	91° × 74°	86° × 56°

Note: The Zenmuse X3 cameranot provide real-time access to the video stream.

detections $\{(b_j, c_j, p_j)\}_{j=1}^{M}$ is outputted. One detection consists of a bounding box $b_j = (x_j, y_j, w_j, h_j)$ defined by its location and size as well as the predicted class $c_j \in \{1, 2, ..., c\}$ and confidence value p_j. Single-stage methods use CNNs to compute many bounding box predictions, which are then filtered by a confidence threshold β to decrease the number of false positives.

A prediction is considered as a true positive (TP) if (i) the predicted class c_p equals the annotated ground truth class c_g and (ii) the intersection over union (IoU) (Equation 2) of the predicted bounding box b_p and ground truth bounding box b_t is larger than a predefined threshold ε. Otherwise, the prediction is a false positive (FP) (Liu et al., 2020).

$$\text{IoU} = \frac{\text{area}(b_p \cap b_t)}{\text{area}(b_p \cup b_t)}. \tag{2}$$

Given TPs, false positives, and false negatives (FNs) (ground truth bounding boxes that have not been predicted), we can define precision $P = \frac{TP}{TP + FP}$ and recall $R = \frac{TP}{TP + FN}$. The harmonic mean between those two metrics is called the F_1 score. $F_1 = \frac{2RP}{R + P}$.

As precision and recall depend on the confidence threshold β, we can generate pairs $(P(\beta), R(\beta))$ when varying β between 0 and 1. Plotting and connecting the points, whose coordinates correspond to these pairs, generates what is referred to as the precision–recall (PR) curve $p_\varepsilon(r)$ that depends on the selected IoU threshold ε. $\text{AP}_\varepsilon = \int_0^1 p_\varepsilon(r)dr$. To obtain a

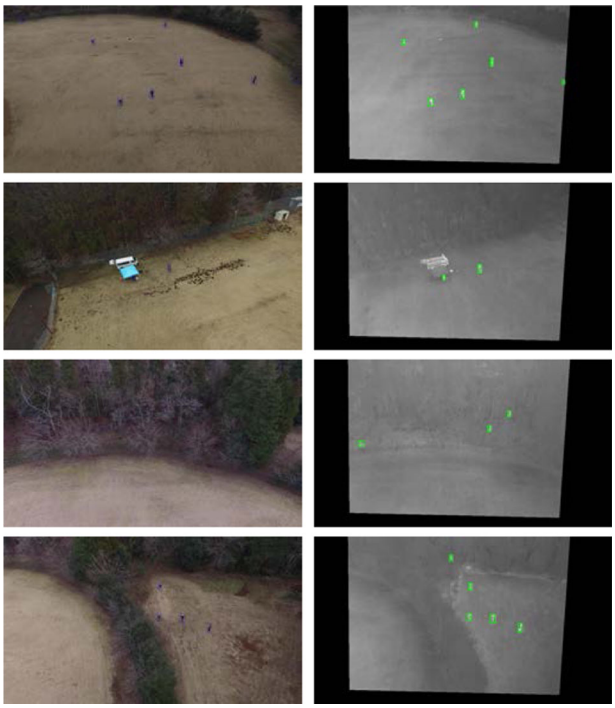

FIGURE 10 Example images of our data set. (Left) RGB images with ground truth labels in blue. (Right) Undistorted FIR images with ground truth labels in green.

single metric from the PR curve, the area under the curve, which is referred to as the average precision (AP), is computed. However, since $p_\varepsilon(r)$ depends on the IoU threshold value ε, it is also denoted as AP_ε. The values $\varepsilon = 0.5$ and $\varepsilon = 0.75$ have been conventionally considered to be standard and strict, respectively. This makes $AP_{0.5}$ and $AP_{0.75}$, which are also denoted as AP@0.5 and AP@0.75, respectively, standard and strict metrics, respectively, the latter being focused on localization.

Since AP depends on the selected IoU threshold ε, an important metric is the AP averaged over multiple IoU thresholds. It is calculated as the average of AP for different ε values, e.g., from $\varepsilon = 0.50$ to $\varepsilon = 0.95$ with a step size of 0.05 (Equation 3).

$$AP = \frac{1}{|S|}\sum_{\varepsilon \in S} AP_\varepsilon \quad \text{where} \quad S = \{0.50, 0.55, \ldots, 0.95\}. \tag{3}$$

To better distinguish the two AP values, we explicitly note which IoU thresholds were used, AP@0.5 for AP_ε and AP@0.05:0.95 for Equation (3), respectively.

Note that mAP refers to the average AP of all object classes, but since we only detect a single class, mAP and AP are the same.

6.2 | Single-spectral models: YOLOv3 and RetinaNet

In this section, we present the findings of training two state-of-the-art DL object detection models, YOLOv3 and RetinaNet. First, we provide an overview of these two neural networks. Subsequently, we compare the two models on our RGB and FIR portions of the NII-CU Multispectral Aerial Person Detection data set. The main aim of the comparison is to determine the model with a lower latency yet acceptable performance. As explained at the end of Section 4.3, the model used for person detection must leave sufficient time and memory to also conduct semantic segmentation.

The YOLO object detector is an accurate one-stage object detector that can achieve low latency. As a feature extractor, YOLO uses Darknet-53, a fully convolutional network with skip connections, as introduced in ResNet. As an alternative, ResNet is also a suitable feature extractor. However, while considerably increasing latency, it only marginally improves detection accuracy (Redmon & Farhadi, 2018). Therefore we only used the Darknet-53 backbone.

The YOLOv3 version used in our study has some improvements over the originally published version by Redmon & Farhadi (2018). We used data augmentation, such as (i) randomly changing the hue, saturation, and value in the HSV color space, (ii) random rotation, translation, scaling, and shearing of the images, and (iii) mosaic data augmentation, which randomly combines four images to one image during training. Those hyperparameters can be optimized with a genetic algorithm during training. Furthermore, we used non-maximum suppression (NMS) for post-processing to filter out duplicated predicted bounding boxes and only output bounding boxes with the highest score.

Many of those improvements were later used in the successor YOLOv4[6] (Bochkovskiy et al., 2020). The main novelty of the RetinaNet[7] object detector, which we used to compare another state-of-the-art object detection model against the YOLOv3 model on our RGB and FIR data sets, is the introduction of the focal loss as described in Section 2.2. Because we aimed to achieve similar runtime compared with YOLO, we selected the faster backbone, namely ResNet-50 resulting in RetinaNet-50. We adjusted the RetinaNet configuration parameters to the ones shown in Table 7 and included smaller anchor ratios and scales to account for our use-case, where people are small objects in terms of pixel size. We used pre-trained weights trained on MS COCO.

Table 8 shows that YOLOv3 with modifications outperformed RetinaNet on both data sets in terms of A*p*@0.5 by 2 percentage points. In our opinion, the reason for the good performance of YOLO is the number of different proven technologies used in the latest YOLO implementations. The number of implemented features in the so-called "bag of freebies" used while training and "bag of specials" used during inference of the YOLO models is higher than in the RetinaNet implementation.

Importantly, the latency of YOLOv3 is only approximately one-third compared with the latency of RetinaNet. Therefore, we selected YOLOv3 as a foundation for further development of multispectral DL models.

[6]The YOLO model we used is based on a publicly available implementation of YOLOv3 in PyTorch with source code available at https://github.com/ultralytics/yolov3

[7]For the RetinaNet model with implementation in Keras we used the source code available at https://github.com/fizyr/keras-retinanet

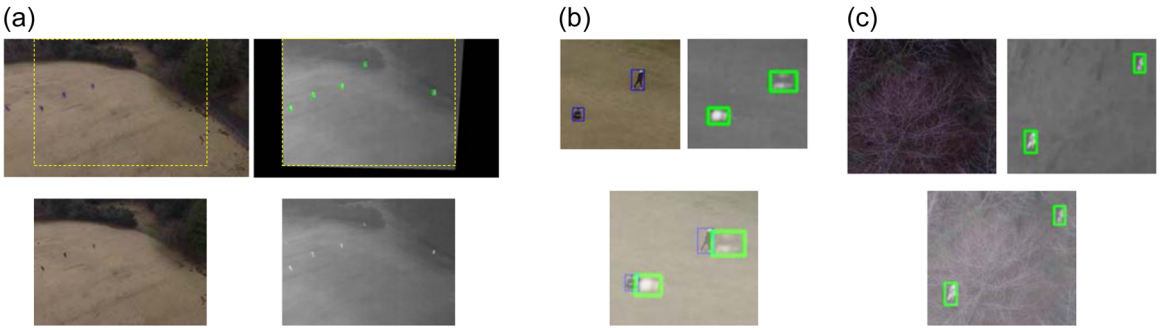

FIGURE 11 (a) Example of a pair of images included in the 4-channel subset of the data set, cropped to the overlapping area between RGB and FIR fields of view. (b) and (c) Examples of images not included in the subset, (b) because the RGB and FIR bounding boxes were misaligned owing to poor time synchronization and rapid drone movement, and (c) because the people were not visible in the RGB image.

TABLE 7 Configuration parameters for RetinaNet and YOLOv3

Hyperparameter	RetinaNet	YOLOv3
Epochs	100	50
Anchor sizes	[16 32 64 128 256 512]	[(10 × 13) (16 × 30) (33 × 23) (30 × 61) (62 × 45) (59 × 119) (116 × 90) (156 × 198) (373 × 326)]
Strides	[8 16 32 64 128]	[8 16 32]
Anchor ratios	[0.3 0.4 1.0 2.0 3.0]	–
Scales	[0.2 0.35 0.5 0.65 1.0 1.2 1.6]	–
Learning rate	$1.0 \cdot 10^{-5}$	$5.79 \cdot 10^{-4}$
Pre-trained weights	resnet50_coco_best_v2.1.0.h5	yolov3-spp-ultralytics.pt

Note: "Steps" indicates the number of batches processed within each epoch. YOLOv3 performed 1245 steps each epoch as all 4980 training images were processed with a batch size of 4.

TABLE 8 Comparison between YOLOv3 and RetinaNet models, using NII's RGB and FIR person detection data sets

Model	AP RGB	AP FIR	lat (ms)
YOLOv3 (Redmon & Farhadi, 2018)	89%	93%	106
RetinaNet (Lin et al., 2017)	87%	91%	310

Note: Average precision (AP) is measured at 0.5 IoU threshold (A*p*@0.5). The latency (lat) of the models in milliseconds are measured on NVIDIA Xavier SoC with resolution of 704 × 416.

6.3 | Multispectral models

In this section, we investigate two possible methods of combining the two input modalities, RGB and FIR, to improve the person detection performance. In Section 6.3.1, we present an approach similar to Takumi et al. (2017), in which an ensemble consisting of two YOLOv3 models is created. In Section 6.3.2, we describe the implementation of a multi-channel input model, which uses aligned 4-channel frames as input.

Our experiments suggested that both models have comparable performance, whereby the multi-channel approach achieves lower latency at a comparable input resolution.

Only the aligned images were used to train and test the multispectral object detection models. An image was considered "well-aligned" if (i) only bounding boxes labeled "both" were present in that image and (ii) no bounding box had the label "bad" in the image.

This procedure reduced the test set to approximately half: 485 images containing 891 labeled people. For this well-aligned data subset, we removed images in which the drone turned rapidly and the RGB and FIR images could not be aligned, as the small time difference between capturing the images resulted in shifted bounding boxes in the direction of the drone rotation. We also removed images in which the annotator only labeled a person in the FIR band, which implied that they were not visible in RGB images (Figure 11). This occurred when people were heavily occluded by trees, resulting in a human being visible only in the FIR band without a clear outline. However, this did not imply that we did not have any images of occluded people

in our "well-aligned" test set, only that the most severe occlusion cases were not tested.

6.3.1 | Ensemble model

We first implemented a multispectral ensemble model similar to Takumi et al. (2017), as promising results were reported with this extreme form of late fusion, also called "decision level fusion" (Konig et al., 2017; Takumi et al., 2017; Wagner et al., 2016).

The ensemble model was constructed from two identical YOLOv3 backbones (Figure 12), trained on the RGB and FIR parts of the NII-CU Multispectral Aerial Person Detection data set, respectively. Both detectors can be trained separately, which means that no matching RGB and thermal data is required for this step. For example, the RGB detector can be pre-trained on MS COCO, which has the benefit of training a more robust and better-performing detector than solely using the KAIST or NII-CU data set. Similarly, to train the FIR detector, we initialized the model with MS COCO pre-trained weights and then trained on our specific FIR data set.

To maintain the network architecture simple, both sub-networks use a $W \times H \times 3$ image as input. Since there is only one channel available in FIR images, we duplicated the channel to have the same number of channels as in the RGB input. As input size for inference, we used the same $W = 960$ and $H = 704$ for FIR and RGB, which can be increased as long as the aspect ratio is unchanged.

With the example of the smallest grid cell layer (output stride = 32), we output a (3, 30, 22, 6) tensor, where 30×22 is the dimension of the grid cells with three predicted bounding boxes for each cell. Each bounding box is defined by six values. All bounding boxes from the RGB and FIR sub-model are concatenated, formulating a (83160, 6) tensor and passed through NMS.

6.3.2 | 4-channel model

The 4-channel model was constructed to use one image with four channels as input (Figure 13). Therefore, the first convolutional layer of YOLO, consisting of 32 filters of size 3×3, was changed to use four channels as inputs. The additional convolutional kernel for each filter was then initialized with the weights of the kernels used to process the red input image channel. The remainder of the model was initialized with weights from a model pre-trained on the MS COCO data set.

To ensure comparability, we trained the model for 50 epochs at a starting learning rate of $6 \cdot 10^{-4}$, using SGD with Nesterov

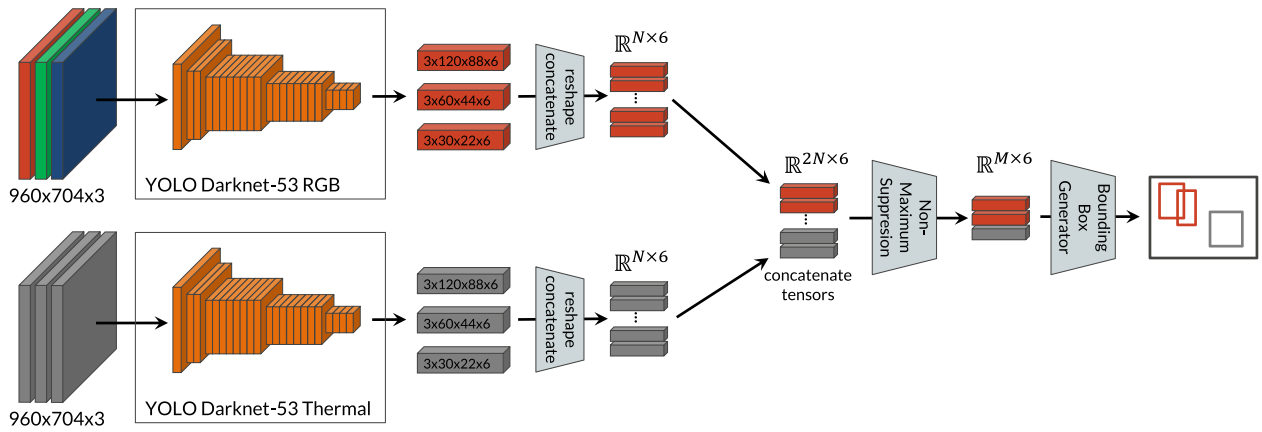

FIGURE 12 Architecture of the ensemble model. Orange building blocks in the YOLO architecture represent the residual blocks within the Darknet-53 backbone network. Compared to the original YOLOv3 implementation, we added the "concatenate tensors" step. Here, we merged predictions from the two individually trained sub-networks. The rest of the detection pipeline was extended to output the information which subnetwork was responsible for one specific bounding box prediction.

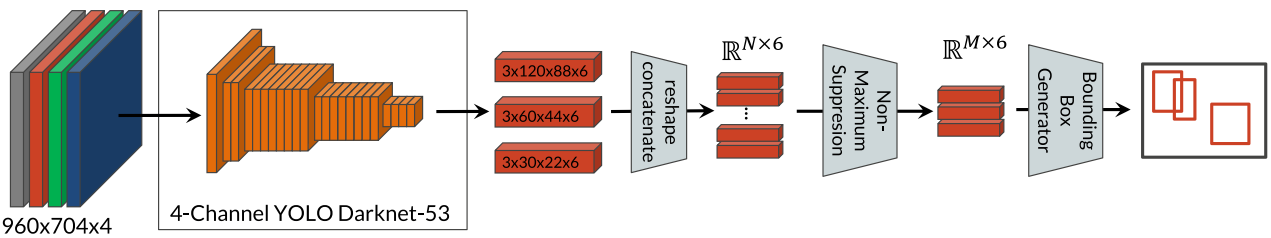

FIGURE 13 Architecture of the 4-channel model. Compared to the original YOLOv3 implementation, we changed the first convolutional layer of YOLOv3. It was modified to take a 4-channel image (RGB and thermal) as input.

momentum and weight decay as the optimizer. The initial learning rate was divided by 10 after the 40th and 45th epoch. We also operated the 4-channel model at an NMS IoU threshold of 0.5 for a fair comparison and did not investigate the further fine-tuning of this value.

Note that in our particular hardware setup, the thermal camera had a smaller field of view compared with the RGB camera (Appendix B, Table 6). Thus, the overlaid thermal and RGB image for the 4-channel model was cropped (Figure 14), such that (1) the resulting image was rectangular with the sides being parallel to the sides of the original RGB image, (2) the resulting image had a RGB and thermal value for each pixel, and (3) the maximal possible rectangle satisfying (1) and (2) was selected.

6.4 | Results for person detection models

The main findings when analyzing the person detection models were as follows (Table 9).

- Overall, the 4-channel model had the best performance of all models, for all IoU thresholds. For the A*p*@0.75 and A*p*@0.5:0.95 metric, the model performed better than the FIR (only) model and significantly better than the ensemble model. The 4-channel model also had a low latency.
- The FIR (only) model was the preferred option when running a single camera setup. It (i) was considerably close to the detection

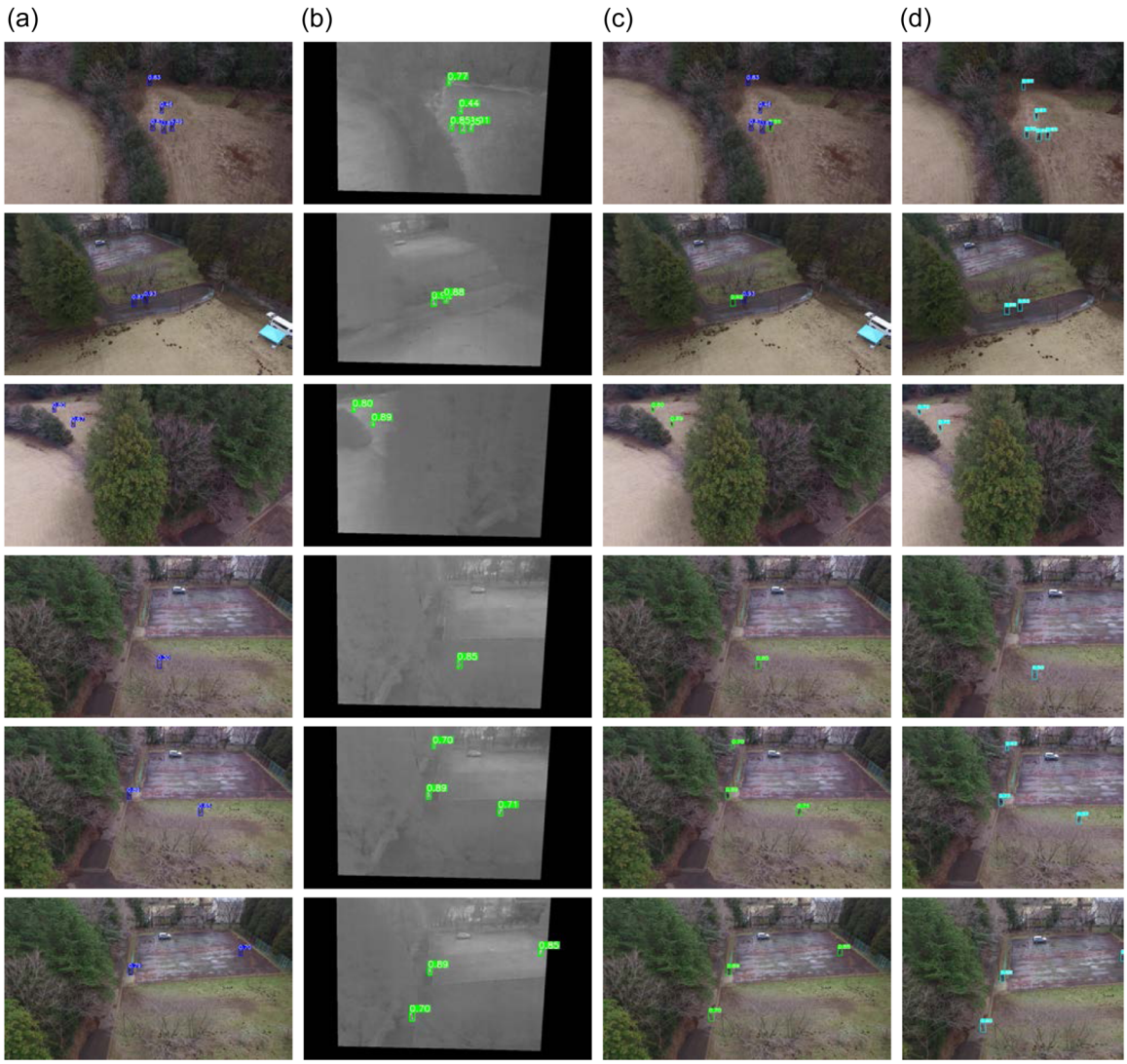

FIGURE 14 (Viewing digitally with zoom is recommended) Detection examples on the NII person detection test set with all four models (RGB, FIR, ensemble, 4-channel) displayed in columns. Boxes in blue color refer to RGB detections, boxes in green color refer to FIR detections, and boxes in cyan color refer to detections from the 4-channel model (we use a new color since we cannot determine which input modality was responsible for the detection). Each row has the same frame as input. The score is displayed on top of each predicted bounding box. Row 1 to 3: easy detection examples, all models detect all visible persons. Row 4: hard detection example, still all models detect the heavily occluded person. Row 5 and 6: RGB model fails to detect one heavily occluded person. However, with the addition of FIR, all persons can be detected. (a) RGB (only), (b) FIR (only), (c) Ensemble, and (d) 4-channel-1920p.

TABLE 9 Comparison between YOLOv3 models on the "aligned" 4-channel validation data set

Model	Ap@0.5	Ap@0.75	Ap@0.5:0.95	lat (ms)
RGB (only)	92.4	44.5	48.3	193
FIR (only)	97.5	72.7	61.7	190
Ensemble	97.3	54.7	53.4	366
4-channel	**97.9**	**76.9**	**64.4**	190

performance of the 4-channel model at A*p*@0.5 and (ii) clearly outperformed the ensemble model. Its simplicity in hardware and software makes it a suitable starting option when developing a drone for monitoring operations.

- In our setup, the ensemble model may not be suitable for an onboard drone mission as (i) it did not excel in any of the metrics and (ii) its latency was almost twice as high as that of the other three models, which also meant higher power consumption per frame.

The ensemble model was considerably close to the 4-channel model in the traditional A*p*@0.5 metric (Table 9). However, when considering the A*p*@0.75 metric, the 4-channel model was 22.2 percentage points better than the ensemble model, amounting to an increase on average precision of 40.6%. Thus, the ensemble model could match the 4-channel model in terms of detection (A*p*@0.5) but had a significantly worse performance for localization (A*p*@0.75). One reason for this inaccuracy of bounding boxes could be that the ensemble model will output the bounding box suggested by either the RGB or FIR subnetwork, yet there is no fusion of bounding box location and dimensions predicted by the RGB and FIR sub-networks. Furthermore, the ensemble model had an almost two times higher latency compared with the 4-channel model (Table 9). Such latency increases the power consumption per frame. The resulting shorter flight time makes the ensemble model impractical for use onboard a drone.

The 4-channel model had a superior IoU averaged AP (A*p*@0.5:0.95) compared with the ensemble and FIR models. Note that the 4-channel model merges the RGB and FIR location information and can output an intermediary bounding box prediction. The improvement of the 4-channel model compared with the FIR only model was 4.2 percentage points, or a 5.8% increase, with A*p*@0.75.

Finally, Figure 15 shows the precision–recall curves for all models. The average precision values (A*p*@0.75) for the displayed curves are shown in Table 9. We can observe that with the 4-channel model, an operating point superior to all other models can be selected. Additionally, the 4-channel model is the only model capable of achieving 100% precision up to a recall value of 6.7%. Its maximum recall value of 85% is considerably close to one possible operating point with precision = recall = 83%.

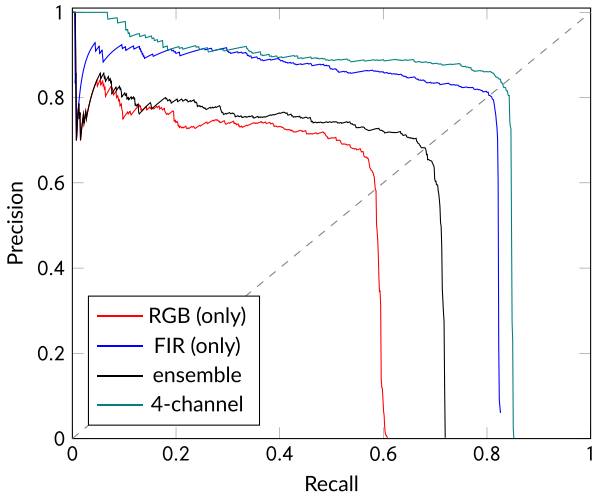

FIGURE 15 Precision–recall (PR) curves for all models at IoU threshold ε = 0.75.

6.5 | Qualitative results

In this section, we present visual results of the four person-detection models. Figure 14 shows some hand-picked detection examples on the NII-CU Multispectral Aerial Person Detection test set. In Figure 16, we display some failure cases of the models, to illustrate the limitations of the models.

The bounding box colors represent the respective models: (blue) detected by the RGB model, (green) detected by the FIR model, (cyan) detected by the 4-channel-1920p model. For the 4-channel model, we cannot distinguish which input modality resulted in detecting the person. Thus, we selected a new color for visualization.

The failure cases in Figure 16 include the following shortcomings:

- In Row 1, the drone rotated rapidly to the left, which caused dis-synchronization (misalignment) of the RGB and FIR frames. All people were correctly detected, yet in the ensemble model, the duplicated bounding boxes were not removed, as their IoU did not reach the specified threshold. To fix this problem, we would need to install cameras that can be triggered to capture pictures at the exact same point in time. Furthermore, during field operation, we can limit the angular velocity of the drone or even optimize the flight path to avoid this problem. However, this data set was recorded when the drone was flown manually and we specifically created this scenario to stress-test our hardware and software. Interestingly, the 4-channel model, which was not trained on such unaligned images, detected all three people in the overlapping vision field with the predicted bounding boxes enclosing the in RGB and FIR unaligned person.
- In Row 2, the RGB model missed a partially occluded person, whereas the thermal model has a false positive prediction caused by trees standing in front of a warm house. This false positive

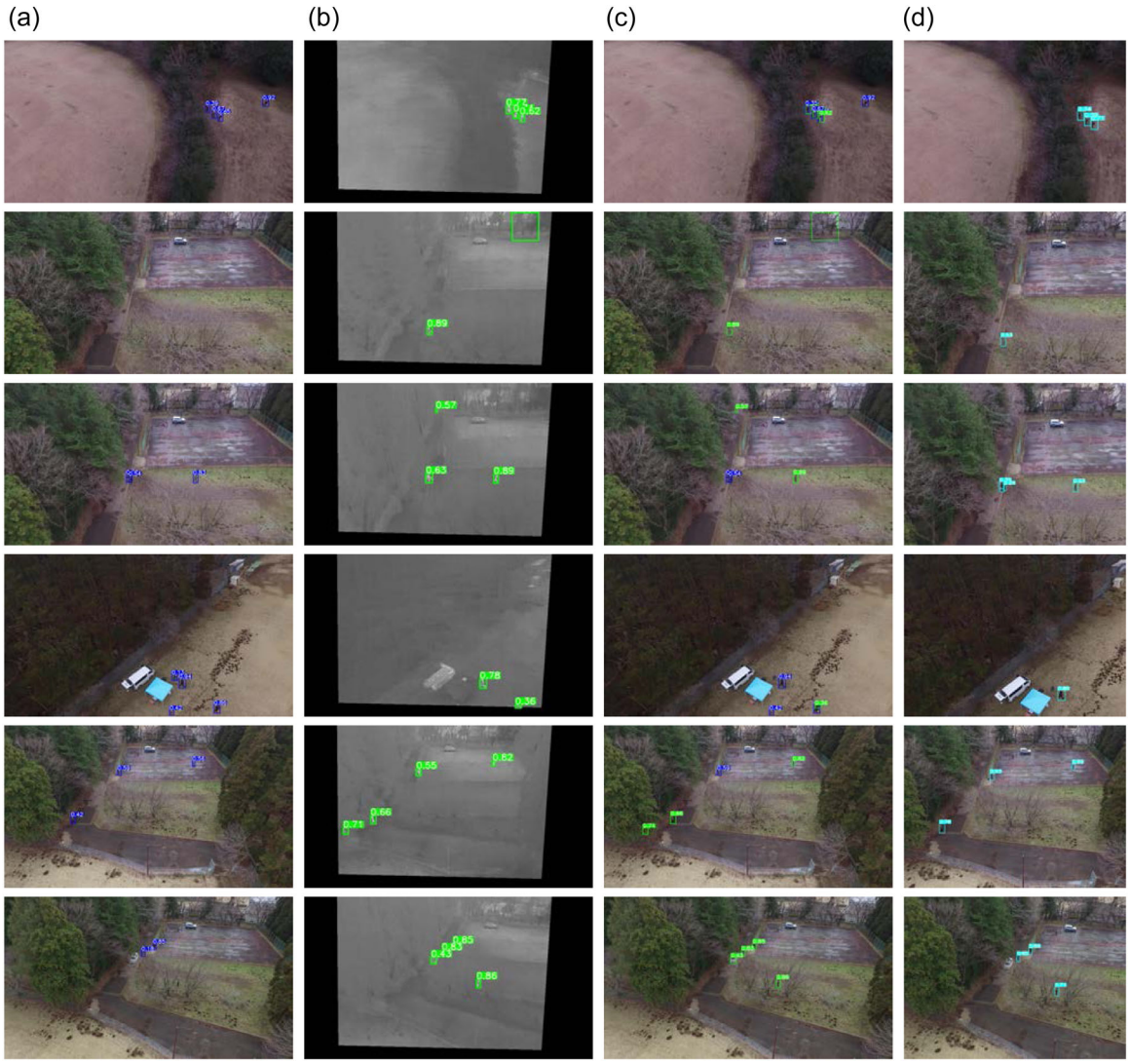

FIGURE 16 (Viewing digitally with zoom is recommended) Continuation of Figure 14. Failure mode examples on the NII person detection test set (a) RGB (only), (b) FIR (only), (c) Ensemble, and (d) 4-channel-1920p.

prediction is also visible in the output of the ensemble model. Since the RGB model is limited in its capacity to detect partially occluded people, we implemented different fusion approaches for both modalities. The false positive detection of the thermal model might be caused by a similar-looking pattern in the training set. In the part of the training set, in which people hid in the woods, only parts of their bodies were observed as warm spots behind the tree pattern. This pattern was similar at the house in the image background. As in Row 1, the 4-channel model predicted the correct output in this scenario, as suggested in Table 9.

- In Row 3, a person held a warm jacket at a distance from their body. This caused duplicated and false positive detections of the jacket in the RGB and 4-channel model. In the FIR model, the bounding box enclosed the jacket, which made the box a bit larger than necessary. The 4-channel model outputs the worst prediction here, displaying three overlapping bounding boxes with comparably low score values. The person in the background was only correctly detected by the FIR and ensemble models. This showed the value of using a FIR camera, as the person could not be easily identified on the RGB image, either by the human labelers or the RGB model. However, even for the jacket special case, the ensemble output was not optimal. In the merging step of the two modalities, the FIR prediction was removed using NMS, as it had a lower score than the correct RGB bounding box. This can only be avoided if the FIR model would output a higher score for the predicted bounding box. In Row 3, the FIR model was the best performing model with the correct amount of tight bounding boxes.
- In Row 4, the RGB model had two false positive detections, a drone on the baseball field and a camping chair. A special type of failure was introduced in the ensemble model, as the FIR input image did not have the same field of view as the RGB image: The FIR model correctly detected the head of a person, as it is the only part of the person it identifies. However, the RGB model also

output a correct bounding box surrounding the entire person. Finally, the failure occurred in the merging step of the ensemble, where the IoU threshold of 0.5 was not reached by both bounding boxes such that none was removed. Such duplicated bounding box detection on the edges of the FIR modality is difficult to solve.

- In Row 5, the ensemble model exhibited superior detection accuracy compared with the 4-channel model. The RGB model did not detect the person standing under a tree on the left side of the image. However, compared with the person in the background in Row 3, this could be detected by a careful human annotator in the RGB modality alone. This person was also missed by the 4-channel model and was the cause for the inferior prediction compared with the FIR and ensemble models.
- In Row 6, the thermal model had a false positive detection at the hot car, whereas the RGB model missed a clearly visible, partially occluded person. The missed person was fixed in the ensemble merging step as the FIR model detected the person. Nevertheless, the false positive car detection could not be corrected. In this example, the 4-channel model operated correctly. However, we also experienced false positive detections in similar conditions with the 4-channel model (house in background, hot car).

7 | CONCLUSIONS

In this paper, we discuss an AI system for a drone that can perform important tasks in disaster relief missions, as specified by the Advanced Robotics Foundation in Japan. We developed a semantic segmentation DL model to support road blockage and landing spot detection. We also developed a person detection DL model. To improve person detection, we created a multispectral data set, consisting of RGB and FIR images. All models were deployed on an onboard mobile device.

The study made three major contributions. *First*, variations of state-of-the-art methods for semantic segmentation were applied to the problem. The performance was evaluated on realistic data sets captured with a drone. The previous segmentation benchmark of 69.7% mIoU was improved to 70.6% mIoU in a comparable setting, using a single model instead of an ensemble. The result of 65.9% mIoU on the threefold cross-validation test set indicated that our selected segmentation model would perform well in the field. To support semantic segmentation studies, we created two data sets publicly available, the "Swiss Drone" data set and the "Okutama Drone" data set. *Second*, we created a novel publicly available data set, the "NII-CU Multispectral Aerial Person Detection" data set, which uses both RGB and thermal (FIR) data and enabled us to design a new object (person) detection model. *Third*, we developed multispectral person detection models. The person detection models combined the RGB and FIR input in two different approaches. The 4-channel model had a significantly higher average precision (increase of 40.6%) for stricter IoU values (0.75) than the ensemble model. Considering all four models (RGB only, FIR only, ensemble, 4-channel), the FIR (only) model was the second best model.

To summarize the lessons learned during this study, we first reflect on the semantic segmentation task of the drone. In a monitoring operation, it is crucial to rapidly scan a large area. This means that the drone must cover a large ground width while operating at high speeds. Therefore, the imaging sensor requires a high resolution. Thus, we selected a 4k imaging sensor. Additionally, our onboard computing device must execute the DL models efficiently. We observed that it is important for the performance of the segmentation task that the images are not downscaled during inference. Therefore, we must use efficient architectures, that is, high mIoU values with low inference times, to cover a large ground patch in the fastest time possible without missing any relevant information.

When considering the person detection task, one of the biggest observations was that the 4-channel architecture was superior in all tasks. We discovered that the early and late fusion approaches differ dramatically in terms of predicting precise bounding boxes. Here, we require further investigation of intermediary fusion approaches such as mid-level fusion or different fusion layer architectures. Additionally, aligning two separate cameras with different lenses on a drone is a complex task as different artifacts, synchronization, and alignment problems must be considered. In practice, we calibrated and rigidly attached the two cameras close to each other on the drone. This enabled us to use a simple method (one homography transformation) to align the images, create a data set, and successfully train models that achieve high accuracy.

Note that the A*p*@0.5 metric can be sufficient when we aim to detect as many humans as possible but are not particular about their exact location. The "localization" metric A*p*@0.75 is appealing for scenarios in which a tight bounding box is required, such as precision dropping of disaster relief supplies for previously detected people.

We observed several limitations in our study. The largest limitation for semantic segmentation is the high latency when performing inference onboard the drone. In the context of a disaster situation, we prefer to process images onboard as network connections may be unavailable, and cloud processing of images is not practical. In a cloud setting, we would not have a trade-off between processing 4k images with the best segmentation performance but a latency of up to multiple seconds or using lower resolution input. When using a lower input resolution, some fine-grained details might become lost. However, a neural network targeted at mobile platforms can reduce latency at the cost of segmentation performance.

An optimal solution may process 4k images (i) in the cloud whenever possible, where the drone only acts as a data recording device, and (ii) onboard the drone with degraded performance or increased latency when the network capacity is limited.

On the technical aspect, as future research, it could be beneficial to experiment with lite reduced atrous spatial pyramid pooling (LR-ASPP), as proposed in Howard et al. (2019), to further reduce the latency of the segmentation models. Training with higher crop sizes on newer hardware generations with more GPU memory will also increase performance. We expect the segmentation metrics to improve when additional compute intensive techniques such as hyperparameter search are used.

The largest limitation for the multispectral person detection model is the lack of infrared data in different environments and for different temperatures. As temperatures increase during the day and alternate during seasons, FIR images may change significantly. For example, we suspect that a model trained on data recorded on a cloudy day in January might not properly generalize a hot summer day in which objects are heated differently by the sun. Additionally, people might become colder than their surroundings, resulting in an inversion of the thermal properties of the image. These observations suggest that solely using FIR images will not result in equally good results in other, unseen environments, particularly when weather conditions differ.

Empirical tests suggest that the models operate best at heights of 35–40 m. Thus, we can cover a sufficiently large area (using the top-down camera) while utilizing the 45° cameras to capture people with sufficient pixel size. At a height of 50 m or more, we observe a significant decrease in performance for person detection. In scenarios in which the drone must fly higher, this could be remedied by using cameras and lenses with a narrower field of view, that is, a higher zoom level. Currently, the person detection model targets "normal" people affected by a disaster who wear "normal" clothes. We plan to create a data set in which people wear special clothes.

We observed that rapid rotations of the drone would occasionally cause the RGB and FIR images to become momentarily unaligned owing to imperfect synchronization between the cameras. This could be fixed by using an external hardware trigger to synchronize the cameras to ensure they operate at the same frame rate.

In our future research, we plan to investigate more methods for fusing RGB and FIR data in addition to our already implemented decision-level fusion (ensemble) and early fusion (multi-channel) approaches. Promising architectures are described in Konig et al. (2017); Liu et al. (2016); Ophoff et al. (2019), where modalities are fused at different positions within the network.

We hope that our system, and the three databases that have been made publicly available, can be an important step toward actual disaster response missions in which drones are used to find people in a large area.

ACKNOWLEDGMENTS

We are grateful to Dr. Raghvendra Jain for his aid with data labeling and his advice on model training. "Research on improving predictability by blending deep learning and symbol processing" was provided to the Graduate School of Engineering, The University of Tokyo under Grant Kakenhi 16H06562.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in NII-CU Multispectral Aerial Person Detection dataset at https://www.nii-cu-multispectral.org, https://www.okutama-segmentation.org/.

ORCID

Simon Speth https://orcid.org/0000-0002-8525-1823
Artur Gonçalves https://orcid.org/0000-0001-8765-7387
Helmut Prendinger https://orcid.org/0000-0003-4654-9835

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C. et al. (2016) Tensorflow: large-scale machine learning on heterogeneous distributed systems. CoRR, abs/1603.04467. http://arxiv.org/abs/1603.04467

Aoki, N., Nishimura, A., Pretto, E.A., Sugimoto, K., Beck, J.R. & Fukui, T. (2004) Survival and cost analysis of fatalities of the Kobe earthquake in Japan. *Prehospital Emergency Care*, *8*, 217–222. https://doi.org/10.1080/312703004386

Azimi, S.M., Henry, C., Sommer, L., Schumann, A. & Vig, E. (2019) Skyscapes fine-grained semantic understanding of aerial scenes. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7393–7403.

Bargoti, S. & Underwood, J. (2017) Deep fruit detection in orchards. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3626–3633.

Bhatnagar, S., Gill, L. & Ghosh, B. (2020) Drone image segmentation using machine and deep learning for mapping raised bog vegetation communities. *Remote Sensing*, *12*, 2602.

Birrell, S., Hughes, J., Cai, J.Y. & Iida, F. (2020) A field-tested robotic harvesting system for iceberg lettuce. *Journal of Field Robotics*, *37*, 225–245.

Bochkovskiy, A., Wang, C.-Y. & Liao, H.-Y.M. (2020) YOLOv4: optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

Chen, J., Wang, G., Luo, L., Gong, W. & Cheng, Z. (2020) Building area estimation in drone aerial images based on mask R-CNN. *IEEE Geoscience and Remote Sensing Letters*, *18*, 891–894.

Chen, L., Zhu, Y., Papandreou, G., Schroff, F. & Adam, H. (2018) Encoder-decoder with atrous separable convolution for semantic image segmentation. CoRR, abs/1802.02611. http://arxiv.org/abs/1802.02611

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K. & Yuille, A.L. (2014) Semantic image segmentation with deep convolutional nets and fully connected CRFs. *arXiv preprint arXiv:1412.7062*.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K. & Yuille, A.L. (2017a) Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*, 834–848.

Chen, L.-C., Papandreou, G., Schroff, F. & Adam, H. (2017b) Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*.

Chen, Y., Xie, H. & Shin, H. (2018) Multi-layer fusion techniques using a CNN for multispectral pedestrian detection. *IET Computer Vision*, *12*, 1179–1187.

Chollet, F. (2017) Xception: Deep learning with depthwise separable convolutions. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1251–1258.

Ciresan, D.C., Meier, U., Masci, J., Gambardella, L.M. & Schmidhuber, J. (2011) Flexible, high performance convolutional neural networks for image classification. In: *Twenty-Second International Joint Conference on Artificial Intelligence*, pp. 1237–1242.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S. & Schiele, B. (2016) The cityscapes dataset for semantic urban scene understanding. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3213–3223.

Daghero, F., Pagliari, D.J. & Poncino, M. (2021) Energy-efficient deep learning inference on edge devices. In: *Advances in Computers*. Vol. 122, pp. 247–301.

Dai, J., Li, Y., He, K. & Sun, J. (2016) R-FCN: object detection via region-based fully convolutional networks. In: *Advances in Neural Information Processing Systems*, pp. 379–387.

Dalal, N. & Triggs, B. (2005) Histograms of oriented gradients for human detection. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. IEEE, pp. 886–893.

Dawdi, T.M., Abdalla, N., Elkalyoubi, Y.M. & Soudan, B. (2020) Locating victims in hot environments using combined thermal and optical imaging. *Computers and Electrical Engineering*, *85*, 106697.

de Oliveira, D.C. & Wehrmeister, M.A. (2018) Using deep learning and low-cost RGB and thermal cameras to detect pedestrians in aerial images captured by multirotor UAV. *Sensors*, *18*, 2244.

Dollár, P., Appel, R., Belongie, S. & Perona, P. (2014) Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*, 1532–1545.

Dollár, P., Wojek, C., Schiele, B. & Perona, P. (2009) Pedestrian detection: a benchmark. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 304–311.

Elkhrachy, I. (2018) Vertical accuracy assessment for SRTM and ASTER digital elevation models: a case study of Najran city, Saudi Arabia. *Ain Shams Engineering Journal*, *9*, 1807–1817. https://doi.org/10.1016/j.asej.2017.01.007. http://www.sciencedirect.com/science/article/pii/S2090447917300084

Endsley, M.R. (1995) Measurement of situation awareness in dynamic systems. *Human Factors*, *37*, 65–84.

Everingham, M., Van Gool, L., Williams, C.K., Winn, J. & Zisserman, A. (2010) The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, *88*, 303–338.

Fu, C.-Y., Liu, W., Ranga, A., Tyagi, A. & Berg, A.C. (2017) DSSD: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*.

Gallego, A.-J., Pertusa, A., Gil, P. & Fisher, R.B. (2019) Detection of bodies in maritime rescue operations using unmanned aerial vehicles with multispectral cameras. *Journal of Field Robotics*, *36*, 782–796.

Geraldes, R., Goncalves, A., Lai, T., Villerabel, M., Deng, W., Salta, A. et al. (2019) UAV-based situational awareness system using deep learning. *IEEE Access*, *7*, 122583–122594. https://doi.org/10.1109/ACCESS.2019.2938249.

Girshick, R., Donahue, J., Darrell, T. & Malik, J. (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587.

Hastie, T., Tibshirani, R. & Friedman, J. (2009) *The elements of statistical learning: data Mining, inference, and prediction*. Springer New York:Springer New York: Springer Science & Business Media.

He, K., Zhang, X., Ren, S. & Sun, J. (2016) Deep residual learning for image recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M. et al. (2019) Searching for MobileNetV3. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1314–1324.

Humamatsu Photonics, K. (2011) *Characteristics and use of infrared detectors*. Hamamatsu City.

Hwang, S., Park, J., Kim, N., Choi, Y. & SoKweon, I. (2015) Multispectral pedestrian detection: benchmark dataset and baseline. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1037–1045.

Johnston, M.G. (2006) Ground object geo-location using UAV video camera. In: *Proceedings of the 25th Digital Avionics Systems Conference*.

Jones, B.F. (1998) A reappraisal of the use of infrared thermal image analysis in medicine. *IEEE Transactions on Medical Imaging*, *17*, 1019–1027.

Kitano, H., Tadokoro, S., Noda, I., Matsubara, H., Takahashi, T. & Shinjou, A. (1999) Robocup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research. In: *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 99CH37028)* Vol. 6. IEEE, pp. 739–743.

Konig, D., Adam, M., Jarvers, C., Layher, G., Neumann, H. & Teutsch, M. (2017) Fully convolutional region proposal networks for multispectral person detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 49–56.

Krajewski, R., Moers, T., Bock, J., Vater, L. & Eckstein, L. (2020) The round dataset: a drone dataset of road user trajectories at roundabouts in Germany. In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, pp. 1–6.

Krizhevsky, A., Sutskever, I. & Hinton, G.E. (2012) Imagenet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems*, pp. 1097–1105.

Langlotz, C.P., Allen, B., Erickson, B.J., Kalpathy-Cramer, J., Bigelow, K., Cook, T.S. et al. (2019) A roadmap for foundational research on artificial intelligence in medical imaging: from the 2018 NIH/RSNA/ACR/the academy workshop. *Radiology*, *291*, 781–791.

Laurmaa, J. (2016) A deep learning model for scene segmentation of images captured by drones. Master's thesis, EPFL, Switzerland and NII, Japan.

Leira, F.S., Johansen, T.A. & Fossen, T.I. (2015) Automatic detection, classification and tracking of objects in the ocean surface from UAVs using a thermal camera. In: *2015 IEEE Aerospace Conference*. IEEE, pp. 1–10.

Lin, D., Ji, Y., Lischinski, D., Cohen-Or, D. & Huang, H. (2018) Multi-scale context intertwining for semantic segmentation. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 603–619.

Lin, T., Maire, M., Belongie, S.J., Bourdev, L.D., Girshick, R.B., Hays, J. et al. (2014) Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312. https://arxiv.org/abs/1405.0312

Lin, T.-Y., Goyal, P., Girshick, R., He, K. & Dollár, P. (2017) Focal loss for dense object detection. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2980–2988.

Liu, J., Zhang, S., Wang, S. & Metaxas, D.N. (2016) Multispectral deep neural networks for pedestrian detection. *arXiv preprint arXiv:1611.02644*.

Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X. & Pietikäinen, M. (2020) Deep learning for generic object detection: a survey. *International Journal of Computer Vision*, *128*, 261–318.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. & Berg, A.C. (2016) SSD: single shot multibox detector. In: *European Conference on Computer Vision*. Springer, pp. 21–37.

Liu, X., Song, L., Liu, S. & Zhang, Y. (2021) A review of deep-learning-based medical image segmentation methods. *Sustainability*, *13*, 1224.

Long, J., Shelhamer, E. & Darrell, T. (2015) Fully convolutional networks for semantic segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440.

Mapzen. (2004) *Nextzen*. Available at: https://www.nextzen.org [Accessed 11th May 2020].

Matsuno, F. & Tadokoro, S. (2004) Rescue robots and systems in Japan. In: *2004 IEEE International Conference on Robotics and Biomimetics*, pp. 12–20.

Nardari, G.V., Romero, R.A., Guizilini, V.C., Mareco, W.E., Milori, D.M., Villas-Boas, P.R. et al. (2018) Crop anomaly identification with color filters and convolutional neural networks. In: *2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE)*. IEEE, pp. 363–369.

Nigam, I., Huang, C. & Ramanan, D. (2018) Ensemble knowledge transfer for semantic segmentation. In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, pp. 1499–1508.

Ophoff, T., Van Beeck, K. & Goedemé, T. (2019) Exploring RGB. depth fusion for real-time object detection. *Sensors*, *19*, 866.

Papageorgiou, C. & Poggio, T. (2000) A trainable system for object detection. *International Journal of Computer Vision*, *38*, 15–33.

Papageorgiou, C.P., Oren, M. & Poggio, T. (1998) A general framework for object detection. In: *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*. IEEE, pp. 555–562.

Pijnacker, H.B.J., Scheper, K.Y. & De Croon, G.C. (2018) Vertical landing for micro air vehicles using event-based optical flow. *Journal of Field Robotics*, *35*, 69–90.

Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. (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.
Redmon, J. & Farhadi, A. (2017) YOLO9000: better, faster, stronger. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7263–7271.
Redmon, J. & Farhadi, A. (2018) YOLOv3: an incremental improvement. *arXiv preprint arXiv:1804.02767*.
Romera, E., Alvarez, J.M., Bergasa, L.M. & Arroyo, R. (2017) Erfnet: efficient residual factorized convnet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, *19*, 263–272.
Ronneberger, O., Fischer, P. & Brox, T. (2015) U-net: convolutional networks for biomedical image segmentation. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer, pp. 234–241.
Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S. et al. (2015) Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, *115*, 211–252.
Sa, I., Ge, Z., Dayoub, F., Upcroft, B., Perez, T. & McCool, C. (2016) Deepfruits: a fruit detection system using deep neural networks. *Sensors*, *16*, 1222.
Saha, P. & Mukhopadhyay, S. (2020) Multispectral information fusion with reinforcement learning for object tracking in IoT edge devices. *IEEE Sensors Journal*, *20*, 4333–4344.
Salamí, E., Barrado, C., Pastor, E., Royo, P. & Santamaria, E. (2013) Real-time data processing for the airborne detection of hot spots. *Journal of Aerospace Information Systems*, *10*, 444–450.
Schedl, D.C., Kurmi, I. & Bimber, O. (2020) Search and rescue with airborne optical sectioning. *Nature Machine Intelligence*, *2*, 783–790.
SenseFly. (2016) *Example datasets: sensefly sa*. Available at: https://www.sensefly.com/education/datasets/ [Accessed on 13th May 2020].
Sergio, G. & Nathan S. (2016) *TensorFlow-Slim: a lightweight library for defining, training and evaluating complex models in TensorFlow*. Available at: https://github.com/google-research/tf-slim [Accessed 29th June 2019].
Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R. & LeCun, Y. (2013) Overfeat: integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*.
Simonyan, K. & Zisserman, A. (2014) Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
St-Laurent, L., Maldague, X. & Prévost, D. (2007) Combination of colour and thermal sensors for enhanced object detection. In: *2007 10th International Conference on Information Fusion*. IEEE, pp. 1–8.
Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z. (2016) Rethinking the inception architecture for computer vision. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826.
Tadokoro, S. (2009) *Rescue robotics: DDT project on robots and systems for urban search and rescue*. Springer London:Springer London: Springer Science & Business Media.
Takumi, K., Watanabe, K., Ha, Q., Tejero-De-Pablos, A., Ushiku, Y. & Harada, T. (2017) Multispectral object detection for autonomous vehicles. In: *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*, pp. 35–43.
Tan, M. & Le, Q.V. (2019) EfficientNet: rethinking model scaling for convolutional neural networks. In: *Proceedings of the 36th International Conference on Machine Learning*.
Tan, M., Pang, R. & Le, Q.V. (2020) EfficientDet: scalable and efficient object detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
Tsukiyama, T. & Shirai, Y. (1985) Detection of the movements of persons from a sparse sequence of TV images. *Pattern Recognition*, *18*, 207–213.
US Geological Survey (USGS). (2020) *Earthquake Hazards Program—Earthquake Catalog*. Available at: https://earthquake.usgs.gov/earthquakes/search/ [Accessed 16th April 2020].
Wagner, J., Fischer, V., Herman, M. & Behnke, S. (2016) Multispectral pedestrian detection using deep fusion convolutional neural networks. In: *Proceedings of the 24th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*.
Wosner, O., Farjon, G. & Bar-Hillel, A. (2021) Object detection in agricultural contexts: a multiple resolution benchmark and comparison to human. *Computers and Electronics in Agriculture*, *189*, 106404.
Yu, H., Li, G., Zhang, W., Huang, Q., Du, D., Tian, Q. & Sebe, N. (2020) The unmanned aerial vehicle benchmark: object detection, tracking and baseline. *International Journal of Computer Vision*, *128*, 1141–1159.
Yuan, J., Gleason, S.S. & Cheriyadat, A.M. (2013) Systematic benchmarking of aerial image segmentation. *IEEE Geoscience and Remote Sensing Letters*, *10*, 1527–1531.
Zhao, H., Shi, J., Qi, X., Wang, X. & Jia, J. (2017) Pyramid scene parsing network. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2881–2890.

How to cite this article: Speth, S., Gonçalves, A., Rigault, B., Suzuki, S., Bouazizi, M., Matsuo, Y. et al. (2022) Deep learning with RGB and thermal images onboard a drone for monitoring operations. *Journal of Field Robotics*, 39, 840–868. https://doi.org/10.1002/rob.22082

APPENDIX A: DISASTER RELIEF

The following description can be found in the documents of the Advanced Robotics Foundation.[8]

Mission overview

A large-scale earthquake has happened. There is a possibility that a serious disaster has happened in a remote village, but details are unclear. It is necessary to confirm the damage status and dispatch a rescue team appropriately according to the situation. However, landslides and other obstructions on various routes obstruct the path to the site, meaning that a secure route to the site must be formulated first. [...]

Mission Phase 1: Formulating a route using aerial search

Formulate a route to the disaster site and search for people in need of help.

An approximately 25 square kilometers search area will be specified in advance. Teams will search the area using an aerial robot. Various routes to the disaster site will be present with this area but maybe obstructed by landslides and other obstructions caused by the earthquake. There are obstacles that cannot be removed immediately (such as large-scale landslides) and obstacles that can be removed (small fallen trees, driftwoods, etc.). Each team will identify the location and type of obstructions, formulate a route to the disaster site by searching with a flying robot and report to the disaster response HQ as quickly as possible.

[8] https://arf.or.jp/?p=1105%26lang=en

There may also be one or more people in need of help rescuers in the area, such as a person involved in fallen trees. The accurate locations of these people must be reported. Ideally, this reporting will be provided in the form of wide-area electronic data such as orthophotos or 3D maps.

Obstructions may use actual sand, trees, etc., or maybe designated using specified markers. People in need may be actual people, or represented by mannequins. In either case, teams are encouraged to use AI-based photo recognition and marking. If using these methodology, teams must conduct this analysis using photos taken by their aerial robot, and must also submit any images produced by this analysis. [...]

APPENDIX B: META CAMERA

The Meta Camera component receives the images of the three cameras and the drone telemetry, synchronizes them, aligns the Thermal image with the 45° RGB image, and makes the images +metadata available for other components in the system, such as the DL components. This section explains in detail the hardware used and the software implementation.

For person detection, we use a FLIR Vue Pro thermal camera and an e-con Systems See3CAM CU135 RGB camera (Figure 4); the full specs of both cameras are shown in Table 6. The RGB camera has a larger field of view than the thermal camera, which allows us to align the images without cropping any part of the thermal image. To increase the RGB camera's field of view, we use a 4:3 aspect ratio resolution (2880 × 2160) instead of 16:9 (3840 × 2160). Both cameras are mounted on a gimbal pointing 45 degrees down. We added an IR cut-off filter to the RGB camera lens, which cuts off all light above a cut-off wavelength of 710 nm. This filter is needed as a typical CMOS image sensor is also responsive to light in the near-infrared spectrum. We also mounted a lens hood on each RGB camera to block lens flares and reflections. The thermal camera was connected to the NVIDIA AGX Xavier with a Macrosil AV to USB2.0 analog video adapter.

The images of the three cameras were undistorted in OpenCV, using their intrinsic camera matrix and lens distortion coefficients, and the thermal image was projected on top of the 45-degree RGB image using a planar homography. This aligned the two images, whereby the larger RGB image remained unchanged. We used OpenCV to estimate the camera parameters, as described in https://docs.opencv.org/master/dc/dbb/tutorial_py_calibration.html. Specifically, we used **cv.findChessboardCorners** to detect the corners of the calibration pattern, and **cv.calibrateCamera**, which outputs the 3 × 3 camera matrix and the 5 × 1 distortion coefficients vector, of which three parameters represent radial distortion, and two represent tangential distortion. To calibrate the thermal camera, we made our own calibration pattern using a metallic plate and foam squares. The metallic plate was heated just before calibration to generate the contrast in the FIR image.

For the person detection application, we also performed the extrinsic calibration by estimating the homography transformation between the RGB and Thermal camera, which were both mounted at 45°. This transformation let us align the two images by warping the Thermal image. This was done with Algorithm 3 and is illustrated in Figure B1. In general, a homography does not mathematically describe the transformation between images of two cameras, but in our case, the distance between the cameras and the objects of interest (people) was much larger than the distance between the cameras (interocular distance), so approximating the transformation with a homography was reasonable. Specifically, with our data collection camera setup (interocular distance 5.4 cm between the FLIR thermal camera and eCon Systems RGB camera), if the homography was estimated for objects at 20 m, but then the system was used for objects at 50 m, the parallax error would be 0.092°, or 2.7 pixels in the RGB image, or 0.8 pixels in the thermal image. As we show in Figure 9b, person height is much larger than 2.7 pixels, so we considered this error to be negligible. The interocular distance between the FLIR and eCon Systems cameras used for the onboard setup was even smaller, at 4.8 cm, so the error would be smaller too.

To synchronize the camera feeds, our onboard application captures and buffers up to four frames from each camera, together with the timestamp from the system clock for each frame. When the DL models request frames to perform inference on, the application finds the pair of RGB/IR frames with the smallest difference of timestamps and returns that pair of frames. This is necessary because the RGB camera and the analog video adapter of the thermal camera have different frame rates. The frames are also synchronized with the telemetry coming from the drone's flight controller.

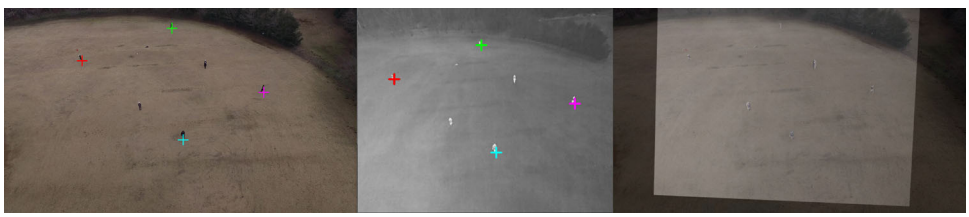

FIGURE B1 Example of calculation of the transformation between the two cameras, using four pairs of matched points. Left: undistorted RGB; center: undistorted thermal; right: aligned thermal overlaid on RGB. The reprojection error, measured on the two people without points, is around five pixels. The image dimensions are 3840 × 2160.

Algorithm 3 Estimate homography between RGB and Thermal camera

```
function GetHomography(rgbImage, thermalImage, thermalCamMatrix, thermalDistortionCoeffs, rgbPoints, thermalPoints)
Require: Images, camera parameters, and two lists of 4 matched pairs of points
    rgbImage = cv2.undistort(rgbImage, rgbCamMatrix, rgbDistortionCoeffs)
    thermalImage = cv2.undistort(thermalImage, thermalCamMatrix, thermalDistortionCoeffs)
    rgbScale = thermalImage.height / rgbImage.height        ▷ The Thermal camera has a lower resolution, so we scale down the RGB points
    for all p ∈ rgbPoints do                                ▷ This is done so the homography will not scale up the thermal image
        p = p * rgbScale
    end for
    Homography = cv2.getPerspectiveTransform(thermalPoints, rgbPoints)   ▷ OpenCV computes a 3x3 matrix corresponding to the homography
    return Homography
end function

procedure Align(rgbImage, thermalImage, thermalCamMatrix, thermalDistortionCoeffs, homography)
    rgbImage = cv2.undistort(rgbImage, rgbCamMatrix, rgbDistortionCoeffs)
    thermalImage = cv2.undistort(thermalImage, thermalCamMatrix, thermalDistortionCoeffs)
    rgbAspectRatio = rgbImage.width / rgbImage.height;      ▷ Pad Thermal image to match aspect ratio of the RGB image
    width = thermalImage.height * rgbAspectRatio;
    height = thermalImage.width;
     thermalImageAligned =
    cv2.warpPerspective(thermalImage, homography, (width, height))   ▷ Warp thermal image so it aligns with the RGB image, and output with
specified dimensions
end procedure
```

APPENDIX C: GEO-LOCATION COMPONENT

The Geo-location component is based on Johnston (2006) and Geraldes et al. (2019)'s studies, and follows these steps of computation:

- Correct pixel coordinate for lens distortion.
- Convert pixel to a point in camera coordinates.
- Project a line ('line-of-sight') from the camera center passing by the camera point, and get the intersection of this line with the ground.
- Convert to NED (North-East-Down) coordinates.
- Convert to geographic coordinates.

We improved Step C, where we take the terrain elevation into account, following the method reported by Salamí et al. (2013). The other steps are identical to what was described by Geraldes et al. (2019). The entire algorithm is shown in Algorithm 4.

In addition, we implemented the inverse operation, which takes a geographical coordinate and projects it into the image space. This was done by inverting each step in reverse order, most of which involve inverting rotation matrices. This was used to check if known objects, such as roads, were visible in a certain image, and where they should be located.

We obtained the terrain elevation from Nextzen Mapzen (2020) Terrain Tiles service, which stores elevation heightmaps as 256 × 256 PNG images. We used zoom level 14, which corresponds to a horizontal resolution of approximately 7.5 meters per pixel, and the vertical precision is ±16 m, according to the SRTM data specification Elkhrachy (2018). Because we have access to the drone's height relative to the take-off position, we can cancel out systematic errors in the elevation data, and Elkhrachy (2018) reports that the SRTM precision is better than the official value of 16 m in certain areas. In practice, we found our approach to work well in the real world. To avoid artifacts, we use bilinear interpolation when reading elevation from Nextzen images. The images are downloaded and stored on the onboard computer before the flight, for a given target area. Implemented in Python running on the AGX Xavier, the Geo-location method takes 1.24 ms of computation time.

For the terrain elevation handling, we based our code on the pseudo-code shown in fig. 6 of Salamí et al. (2013)'s study. We adapted their stop condition to also stop if two elevation values are approximately equal with a tolerance of 1 mm because our elevation map's bilinear interpolation causes the returned values to always differ slightly and these tiny differences would lead to extra iterations which brought no tangible added precision.

Algorithm 4 Geo-location algorithm

function getGeolocation($camMatrix, distortionCoeffs, lat_{UAV}, lon_{UAV}, alt_{UAV}, yaw_{UAV}, pitch_{cam}, p$)

Require: Camera parameters, UAV telemetry, camera angle, and a point p in pixels. If p was picked from an already-undistorted image, then $distortionCoeffs$ should be all zeros.

Require: Camera is in a gimbal, so camera pitch is constant and roll is zero

$\mathbf{p}_{cam}$ = cv2.undistortPoints($\mathbf{p}, camMatrix, distortionCoeffs$) ▷ Convert from image to camera space, using OpenCV

$\mathbf{R}_{cam}^{ground}$ = GetRotMat(0, 0, 90 − $pitch_{cam}$)$^{-1}$ ▷ Get projection in imaginary flat ground. Ground plane is defined by

$\mathbf{gn}_{cam} = \mathbf{R}_{cam}^{ground} \begin{bmatrix} 0, 0, 1 \end{bmatrix}^T$ ▷ ...a normal vector pointing up, and

$\mathbf{gp}_{cam} = \mathbf{R}_{cam}^{ground} \begin{bmatrix} 0, 0, -alt_{UAV} \end{bmatrix}^T$ ▷ ...a point directly below the drone, in camera space

$dist = \frac{\mathbf{gp}_{cam} \cdot \mathbf{gn}_{cam}}{\mathbf{p}_{cam} \cdot \mathbf{gn}_{cam}}$ ▷ Distance from camera to flat-ground target

$\mathbf{target}_{cam} = dist \cdot \mathbf{p}_{cam}$ ▷ Flat-ground target, in camera space

$\mathbf{R}_{UAV}^{cam}$ = GetRotMat(0, 0, $pitch_{cam}$)$^{-1}$ ▷ Convert to North-East-Up (NED) coordinates

$\mathbf{R}_{body}^{cam}$ = GetRotMat(-90, 0, -90) $\mathbf{R}_{UAV}^{cam}$

$\mathbf{R}_{NED}^{body}$ = GetRotMat(yaw_{UAV}, 0, 0)

$\mathbf{R}_{NED}^{cam} = \mathbf{R}_{NED}^{body} \mathbf{R}_{body}^{cam}$

$\mathbf{target}_{NED} = \mathbf{R}_{NED}^{cam} \mathbf{target}_{cam}$

$h_t[i = 0]$ = getElevation($\mathbf{uav}_{Geo}$)) ▷ Check intersection with terrain, iteratively (Salamí et al., 2013); start at elevation at UAV position

while isNew($elevation_t[i]$, $eps = 0.001$) and $i < MAX$ **do** ▷ While different elevation values, with tolerance of 1 mm

  $\mathbf{target}_{NED}$ = intersection($z_{NED} = alt_{UAV} - elevation_t[i]$) ▷ Intersect ray line with flat plane at elevation

  $\mathbf{target}_{Geo}$ = NED2Geo($\mathbf{target}_{NED}$) ▷ Convert estimated target to geodetic coordinates

  $elevation_t[i = i + 1]$ = getElevation($\mathbf{target}_{Geo}$)) ▷ Elevation at new estimated target, for next iteration

end while

if !isNew($elevation_t[i]$) and $elevation_t[i] \neq elevation_t[i-1]$ **then** ▷ When there's cyclical repetition of elevations

  $elevation_t[i]$ = average($elevation_t[i], elevation_t[i-1]$) ▷ Average of last different heights

  $\mathbf{target}_{NED}$ = intersection($z_{NED} = alt_{UAV} - elevation_t[i]$) ▷ Intersect ray line with flat plane at elevation

  $\mathbf{target}_{Geo}$ = NED2Geo($\mathbf{target}_{NED}$) ▷ Convert estimated target to geodetic coordinates

end if

return $\mathbf{target}_{Geo}$

end function

function NED2Geo($\mathbf{p}_{NED}$, $\mathbf{ref}_{Geo}$)

Require: A point in NED coordinates, and a reference geographic position (e.g. the UAV position)

$m_1 = 111132.92, m_2 = -559.82, m_3 = 1.175, m_4 = -0.0023, p_1 = 111412.84, p_2 = -93.5, p_3 = 0.118$ ▷ Approximation constants

$\phi = \mathbf{ref}_{Geo,1}$ ▷ Scale between degrees and meters depends on the latitude

$scale_{lat} = m_1 + m_2 \cos(2\phi) + m_3 \cos(4\phi) + m_4 \cos(6\phi)$ ▷ One degree of latitude in meters

$scale_{lon} = p_1 \cos(\phi) + p_2 \cos(3\phi) + p_3 \cos(5\phi)$ ▷ One degree of longitude in meters

$\mathbf{p}_{Geo} = \mathbf{ref}_{Geo} + \begin{bmatrix} \frac{\mathbf{p}_{NED,1}}{scale_{lat}} & \frac{\mathbf{p}_{NED,2}}{scale_{lon}} \end{bmatrix}^T$

return $\mathbf{p}_{Geo}$

end function

function GetRotMat(y, p, r)

Require: Yaw, pitch and roll angles

return $\begin{bmatrix} \cos(y) & -\sin(y) & 0 \\ \sin(y) & \cos(y) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(p) & 0 & \sin(p) \\ 0 & 1 & 0 \\ -\sin(p) & 0 & \cos(p) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(r) & -\sin(r) \\ 0 & \sin(r) & \cos(r) \end{bmatrix}$ ▷ Convert degrees to radians depending on implementation

end function

In Algorithm 4, angles are expressed in degrees and converted to radians where needed (trigonometry functions, etc.). These conversions were omitted here for simplicity. The following notation is used: lowercase variables x are scalars; a subscript x_{foo} is purely descriptive; lowercase bold variables $\mathbf{p}$ are vectors; a subscript $\mathbf{p}_{cam}$ indicates that the vector is in coordinate space *cam*; a numerical subscript $\mathbf{p}_{cam,n}$ denotes the nth component of the vector; uppercase bold variables $\mathbf{R}$ are matrices, and matrices with both subscripts *and* superscripts $\mathbf{R}_b^a$ indicate a rotation matrix from coordinate space a to space b.