remote sensing

MDPI

Article

Autonomous, Onboard Vision-Based Trash and Litter Detection in Low Altitude Aerial Images Collected by an Unmanned Aerial Vehicle

Marek Kraft *, Mateusz Piechocki, Bartosz Ptak and Krzysztof Walas

Faculty of Control, Robotics and Electrical, Engineering, Institute of Robotics and Machine Intelligence, Poznań University of Technology, 60-965 Poznań, Poland; mateusz.piechocki@student.put.poznan.pl (M.P.); bartosz.ptak@student.put.poznan.pl (B.P.); krzysztof.walas@put.poznan.pl (K.W.)
* Correspondence: marek.kraft@put.poznan.pl

Abstract: Public littering and discarded trash are, despite the effort being put to limit it, still a serious ecological, aesthetic, and social problem. The problematic waste is usually localised and picked up by designated personnel, which is a tiresome, time-consuming task. This paper proposes a low-cost solution enabling the localisation of trash and litter objects in low altitude imagery collected by an unmanned aerial vehicle (UAV) during an autonomous patrol mission. The objects of interest are detected in the acquired images and put on the global map using a set of onboard sensors commonly found in typical UAV autopilots. The core object detection algorithm is based on deep, convolutional neural networks. Since the task is domain-specific, a dedicated dataset of images containing objects of interest was collected and annotated. The dataset is made publicly available, and its description is contained in the paper. The dataset was used to test a range of embedded devices enabling the deployment of deep neural networks for inference onboard the UAV. The results of measurements in terms of detection accuracy and processing speed are enclosed, and recommendations for the neural network model and hardware platform are given based on the obtained values. The complete system can be put together using inexpensive, off-the-shelf components, and perform autonomous localisation of discarded trash, relieving human personnel of this burdensome task, and enabling automated pickup planning.

Keywords: deep learning; object detection; image processing; trash; litter; UAV; YOLO

Citation: Kraft, M.; Piechocki, M.; Ptak, B.; Walas, K. Autonomous, Onboard Vision-Based Trash and Litter Detection in Low Altitude Aerial Images Collected by an Unmanned Aerial Vehicle. *Remote Sens.* **2021**, *13*, 965. https://doi.org/10.3390/rs13050965

Academic Editor: Bahram Salehi

Received: 29 January 2021
Accepted: 26 February 2021
Published: 4 March 2021

1. Introduction

Despite numerous efforts to raise awareness of its consequences, public littering is a serious, ongoing problem. Abandoned trash is an issue of significant economic, aesthetic and ecological impact. While the full analysis of littering behaviour is a complex issue beyond the scope of the paper, it is worth noting that it is proven that littering behaviour is much more likely if the environment is already littered. As such, it has a self-perpetuating effect—the presence of litter attracts even more litter. By contrast, a clean community discourages littering [1]. Litter and trash removal from public and natural spaces is mostly done through hand pick up. Such efforts, even when backed up with additional devices engineered to make sanitation workers' work less tiresome and more effective [2], are still considered tedious [3]. Moreover, covering large areas is usually inefficient and relatively costly, as it requires thorough sweeping. Any solution capable of solving the problems at least to some degree is therefore desirable.

In this paper, an unmanned aerial vehicle (UAV) litter and trash detection system is described. Airborne vehicles can observe the terrain from above and freely move with relatively high speed to cover vast areas of land to detect litter. Object detection in aerial images is a much more difficult task than regular object detection. This is mostly due to the relatively small size of detected objects in the image. Moreover, drones can be considered

edge devices, and, if onboard image processing is required, one has to take the limitations of computational power, size, weight, and power consumption into account. It is also not uncommon for such applications to require real-time processing capability.

The main contributions of the paper are as follows: first, a dedicated dataset for aerial rubbish detection is introduced. The dataset, dubbed UAVVaste, contains (at present) 772 images with 3716 hand-labelled annotations of rubbish in the urban and natural environment such as streets, parks, and lawns. The dataset was inspired by Trash Annotations in the Context (TACO) dataset [4], which contains much larger objects from a near-earth perspective, making it less usable for aerial trash detection. The dataset is publicly available in COCO format, including bounding box annotations and segmentation masks. Second, an "eye in the sky" sensor was developed. Based on deep learning object detection and GPS information, the system can detect waste in aerial images and place it on the map with an accuracy, which is sufficient to facilitate easy planning of pickup. Third, a thorough examination of a range of embedded computational platforms was performed. Since the system has to operate onboard and therefore is destined to be mounted on the UAV, the use of limited resource embedded devices is necessary, so an evaluation of selected approaches and optimisation is also included in the paper.

Accurate object detection in images has seen enormous progress over the last few years. The task, though trivial for humans to perform, is still a challenge even for modern computers. However, the introduction of convolutional neural networks (CNNs) has significantly improved accuracy over the classical approaches based on handcrafted features [5–8], pushing the boundaries closer to human-level performance. As a result, the vast majority of modern object detection systems are based on solutions based on CNNs. The first approaches (the RCNN family), which decoupled the detection and recognition process, have evolved gradually to simplify training and inference and integrate the object recognition pipeline into a single neural network, also resulting in a significant speed increase [9–11].

While the Faster-RCNN approach reached the processing speed that was orders of magnitude higher than the RCNN, it is still not suited for real-time applications, especially ones that are not intended to run on high-end GPUs. The need for real-time applicable solutions for object detection based on CNNs has created a wide range of approaches. The first popular solution was YOLO (You Only Look Once) presented in [12]. The approach involves a single neural network trained end-to-end with a joint loss function integrating the components for both the detection and recognition objectives. YOLO forgoes the region proposals stage, framing the detection as a refinement of initial, anchored bounding boxes by regression, while simultaneously predicting their class label. Original YOLO enables processing of images at a rate of over a hundred frames per second, but at a cost of reduced accuracy when compared to the RCNN family. However, the approach gained a lot of traction, and the followup research resulted in solutions that are both fast and accurate, both in and outside the YOLO family [13]. The introduction of architecture improvements such as feature pyramids [14], specialised loss functions such as focal loss [15], or the use of neural architecture search [16] to devise neural networks for better feature extractors [17] and learning data augmentation mechanisms optimised for the detection task [18] have driven the constant progress.

State-of-the-art methods include EfficientDet family of networks [19], which uses a novel implementation of bidirectional feature pyramid network and compound scaling mechanism first introduced in [20]. The more complex variants of EfficientDet are now considered the most accurate based on the mean average precision (mAP) measured on the widespread COCO benchmark dataset [21]. On the lower end, the most recent version of YOLO, named simply YOLOv4, offers the best processing speed versus accuracy trade-off [22]. This is achieved by combining recent advancements such as feature pyramid, spatial attention mechanism, application of modern activation function [23] and advanced augmentation techniques such as random erasing [24,25] and mosaic augmentation, sup-

plemented with self-adversarial training [22] and modern loss function [26]. The most recent YOLOv4 variant also includes the compound scaling [27].

It is clear that the introduction of deep learning enabled dramatic accuracy gains in a wide range of image and video processing applications. However, deep learning is associated with a relatively high computational cost. Streaming high-resolution video in real-time to a remote base station reduces the flexibility and hinders the deployment in many situations due to high bandwidth requirements. Onboard image processing is far more practical but requires a computational platform powerful enough to handle the workload and small and lightweight enough to enable mounting it on the UAV. Power consumption might also be a significant factor since both the additional portion of consumed power and the added weight affect the battery life and therefore shorten the flight time.

Since embedded CPUs are not designed to handle real-time CNN workloads, the presented system is based on dedicated accelerators. The concept of acceleration of computations in image processing and machine learning with dedicated hardware dates back many years [28]. The recent rise of deep learning and associated applications certainly spurred an increase of interest in such solutions. While the first designs were developed mostly for datacenter applications [29,30] or research [31], solutions designed for edge computing were soon to follow [32]. The designs explore a variety of approaches. The majority of accelerators are designed for inference and optimised for low power consumption. Aside from the capability for performing the neural network-related computations in parallel, they often employ reduced precision number representation, some method of quantisation [33] or even binarisation [34] of weights and activations. Quantisation is an especially attractive option, since quantised neural networks oftentimes perform at a level similar to their non-quantised counterparts. At the same time, their implementation is more hardware-efficient, which also drives the manufacturing cost and power consumption down [35]. Model compression by quantisation, model pruning and compression is an area of active research, with numerous novel ideas proposed [36]. On the other hand, quantisation might be hard to apply to some of the more complex neural network models. Since application of deep neural networks in edge computing devices is an area with great market potential, and the best ideas are expected to find their way into market eventually. The system described in this paper, including the specialised computational platform, is based on off-the-shelf components. The Pixhawk autopilot hardware [37,38] with an external Global Positioning System/Global Navigation Satellite System (GPS/GNSS) module with an integrated barometric altimeter is used for UAV control and continuous position monitoring. At the same time, the visual information processing is performed using the dedicated computational platform. Three variants of the computational platform were evaluated—on the higher end of the cost spectrum (but still affordable), the high-performance Nvidia Xavier NX module [39], while the more economic options included the Raspberry Pi 4 computer [40] with deep learning accelerators connected using the USB interface [41].

Deep learning object detection with UAVs is an attractive application, and, as such, it attracts the interest of many researchers. A large portion of the presented solutions relies on onboard processing [42]. However, the applications focus mostly on the detection of objects such as persons and vehicles [43–45]. The characteristics of aerial object detection—mostly, the small apparent size and specific viewpoint—make the object detection problem particularly hard. Particularly dealing with the former is a field of active research [46]. Moreover, the distinctive characteristics of the problem raise the need for field-specific datasets to be used for algorithm training and evaluation. The datasets, with examples being the Stanford Drone Dataset [47] or the VisDrone dataset [48], also focus mostly on person and vehicle detection. The datasets can also vary in terms of context; for example, Ref. [49] presents a dataset aimed at the detection of humans in the wilderness for the purpose of search and rescue missions. In [50], a marine litter monitoring approach using for beaches using drones is presented. The approach is based on personnel reviewing the acquired images. The paper focuses mostly on the operators' performance on a variety of conditions (flight altitude, location, lighting conditions) and the amount of required personnel training.

System for litter and trash detection for coastal areas are described e.g., in [51,52], with the first being human-operated with the purpose for longitudinal study, and the latter relying on cloud resources for processing. Marine litter detection problem is also brought up in [53], but the paper focuses on data generation and deals with underwater detection, while [54] deals with trash detection in water bodies using drones. The latter also introduces an application-specific dataset (AquaTrash), but it is of relatively small size.

2. Materials and Methods

For the purpose of this research, we introduce a new, application-specific dataset. The UAVVaste dataset consists to date of 772 images and 3716 annotations. The main motivation for the creation of the dataset was the lack of domain-specific data. The datasets widely used for object detection evaluation and benchmarking such as the COCO dataset [21] typically contain objects such as vehicles, animals, and household objects devices whose relative size is on average larger than ones found in aerial imagery. Datasets containing trash contain mostly images taken from the point of view of a pedestrian [4] or are small in size [54]. The comparison of relative object size distributions in the TACO and UAVVwaste datasets is shown in Figure 1. The difference is obvious and suggests significant distribution shift between the two domains.

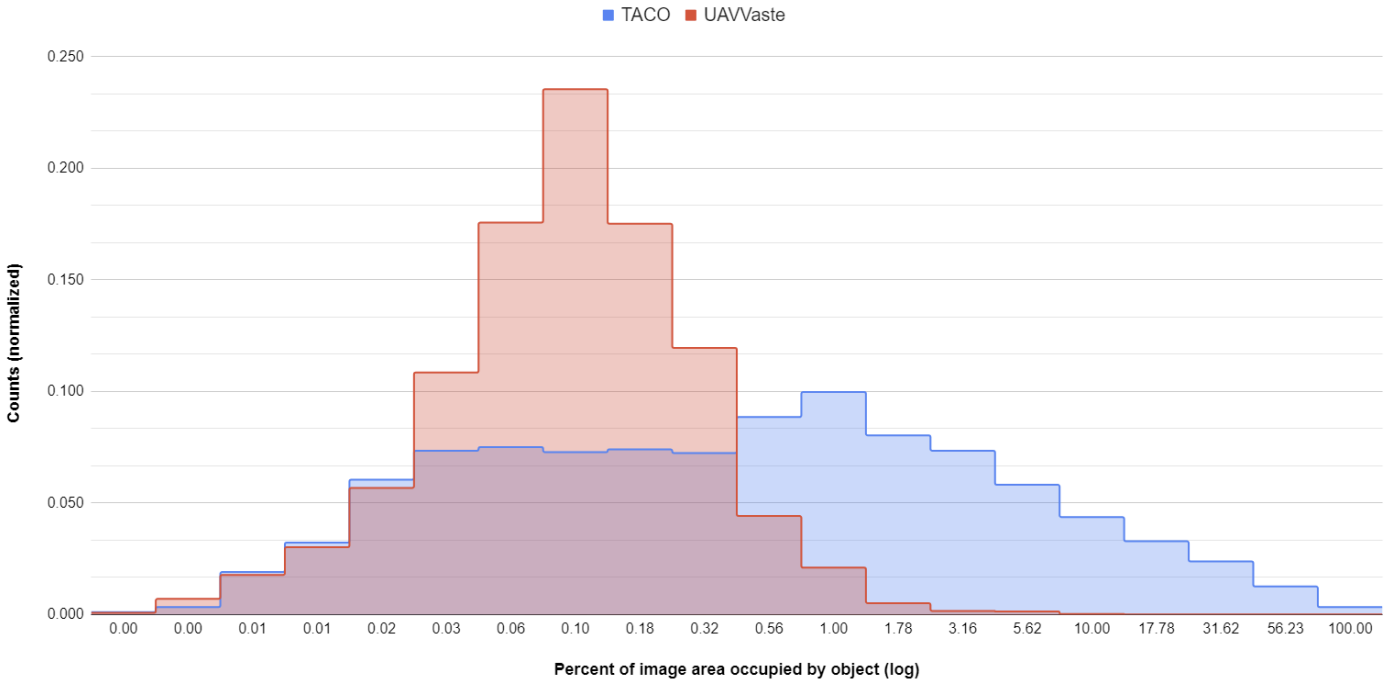

Figure 1. Comparison of relative object size distributions in the TACO and UAVVwaste datasets—Normalized counts versus percent of image occupied by objects. Please note the log-scale on the horizontal axis.

Since general automated UAV-based trash detection and localisation is a promising application, the authors hope that the collected data will facilitate this field's development. The dataset is made publicly available (https://github.com/UAVVaste/UAVVaste, accessed on 1 March 2021) and is intended to be expanded. Sample images and annotations from the dataset are given in Figure 2. As the largest dataset representative for the described application, UAVVaste was used in the evaluations whose results are included in the paper.

Figure 2. Sample images from the UAVVaster dataset with objects of interest highlighted. Please note the variability of backgrounds—both in terms of domain and in terms of the presence of disctractors, and the relatively small size of the objects to be detected.

The UAV hardware is composed of the Pixhawk 2 autopilot controller with an inertial measurement unit providing the compass functionality connected to the compatible Here2 GPS/GNSS module. The module is additionally fitted with a barometric pressure sensor providing the relative altimeter functionality. The parts make it possible to carry out the patrol mission according to a programmed path defined by waypoints, as implemented in the autopilot's basic functionality. Additional components include a downward-facing camera (assured by using a gimbal) and a computational platform. The general block diagram of the system is given in Figure 3. Example configuration with a Nvidia Xavier NX is shown in Figure 4.

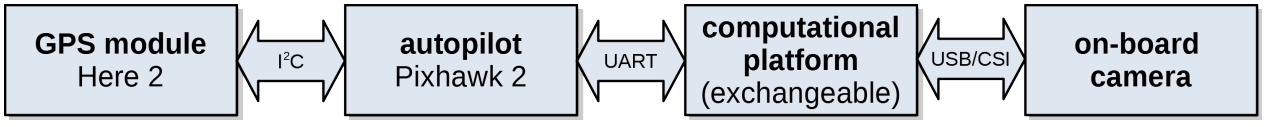

Figure 3. Block diagram of the system with all the essential blocks and interfaces.

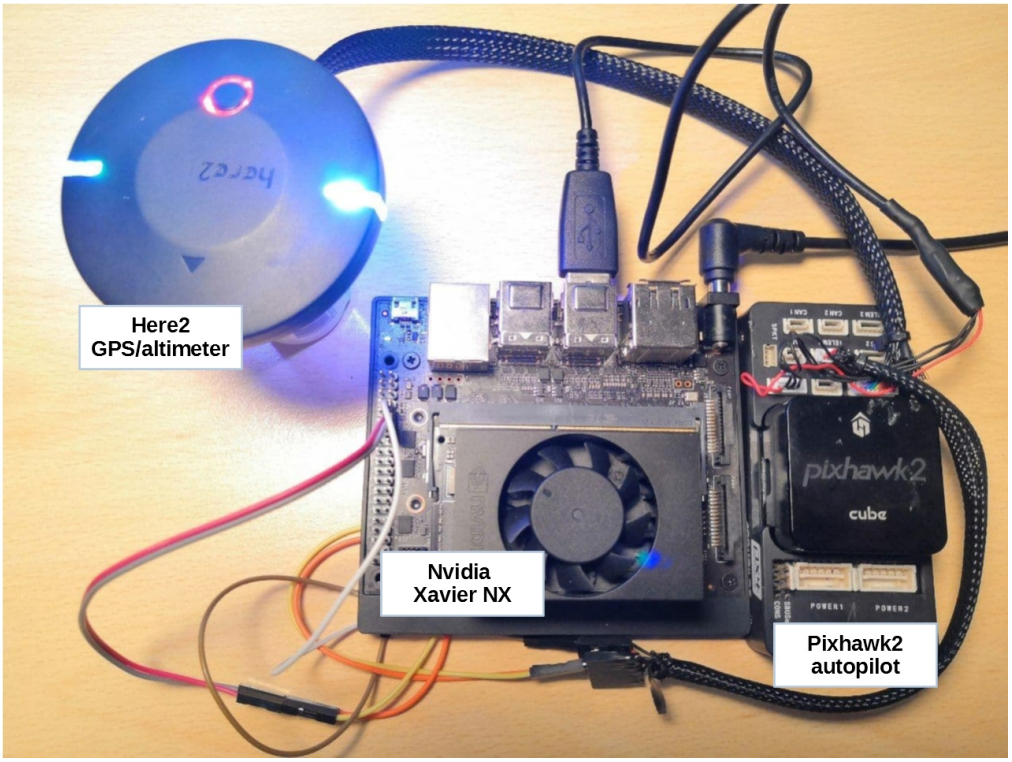

Figure 4. Example configuration of the system with the Nvidia Xavier NX computational platform.

Power consumption, small dimensions, and low weight are essential for UAV-mounted data processing units. With this in mind, the following computational platforms were evaluated:

Nvidia Xavier NX is a system-on-chip designed for embedded applications, integrating multi-core embedded 64-bit CPU and an embedded GPU in a single integrated circuit [39]. The CPU is a six-core, high-performance 64-bit Nvidia Carmel ARM-compatible processor clocked at up to 1.9 GHz (with two cores active, 1.4 GHz otherwise). The GPU is based on Volta architecture, with 384 CUDA cores. Moreover, it is equipped with 64 tensor cores and two Nvidia deep learning accelerator (NVDLA) engines dedicated for deep learning workloads. The CPU and the GPU both share the system memory. The development platform is fitted with 8 GB low power DDRL4 RAM clocked at 1600 MHz with a 128-bit interface, which translates to 51.2 GB/s bandwidth.

Google Coral USB (Tensor Processing Unit) is an accelerator for edge processing based on the Google Tensor Processing Unit (TPU) architecture. The original design is an application-specific integrated circuit (ASIC) designed to be used in datacenters, with machine learning training and inference workloads, with the goal of optimizing the performance to power consumption ratio. Coral is dubbed the edge TPU and is a scaled-down version for embedded applications. An essential part of the edge TPU is the systolic matrix coprocessor with the performance of 4 TOPS, along with a tightly coupled memory pool. It is an inference-only device, and the neural network model needs to be compiled with dedicated tools enabling quantisation in order to be executed on this architecture. The accelerator connects to the host system by either USB (the option used in the presented research) or PCIe interface [41].

Neural Compute Stick 2 (NCS2) is a fanless device in a USB flash drive form factor used for parallel processing acceleration. It is powered by a Myriad X VPU (vision processing unit) that can be applied to various tasks from the computer vision and machine learning domains. The main design goal was to enable relatively high data processing performance in power-constrained devices. This makes the chips a natural choice for powering the devices destined for edge processing. The high performance is facilitated by using 16

SHAVE (Streaming Hybrid Architecture Vector Engine) very long instruction word (VLIW) cores within the SoC, and an additional neural compute engine. Combining multi-core VLIW processing with ample local storage for data caching for each core enables high throughputs for SIMD workloads. It is directly supported by the OpenVINO toolkit, which handles neural network numerical representation and code translation [41,55].

The USB deep learning accelerators were mounted on the Raspberry Pi 4 single-board computer with a quad-core ARM Cortex A72 processor and 4 GB RAM. Since the focus with accelerators is usually high performance to power consumption ratio, they are not necessarily the fastest devices available. Moreover, due to architectural trade-offs, some of the neural network architectures might not be running very effectively, or even be impossible to implement on such devices due to limited programmability or the limitations imposed by mandatory conversion. With this in mind, a range of neural network architectures and variants was tested:

- SSD detector with lightweight backend implemented in TensorFlow Lite [56] was included in the evaluation since it is one of the algorithms that operate without issues on Google Coral. The model uses quantization to 8-bit integers.
- YOLOv3 and YOLOv4 were tested, along with their lightweight ("lite") variants to provide a range of solutions, since architectures with the best accuracy may not be fast enough for real-time processing, especially on resource-limited architectures.
- Models compiled for Nvidia Xavier NX were optimized with TensorRT [57] with the full precision floating point (FP32) variant, half-precision (FP16) variant and, quantized 8-bit variant where possible for potential performance gains. Performance trade-offs in terms of accuracy were also investigated.
- Tests performed using Xavier NX included analysis for a range of batch sizes to assess data transfer operations' impact on the frame rate, with the highest frame rate presented as the evaluation's result. USB accelerators do not operate on batches, so the images were sent one by one.
- Models used with the NCS2 were compiled using the OpenVINO toolkit [55], which translates the code to this architecture and performs conversion of weights and activations to the FP16 format.
- For comparison, the models were also executed on the microprocessors of the Raspberry Pi 4 and the Nvidia Xavier NX platforms. The tested models were based on the TensorFlow Lite implementation [56].
- Both USB accelerators were tested using the USB 2.0 and USB 3.0 ports of Raspberry Pi. The Coral accelerator was tested with two variants of the *libedgetpu* library—The *std* variant for regular clock speed and the *max* variant with higher clock speed, resulting in potentially higher performance. The higher clock frequency might require additional cooling for stable operation.

The training procedures were performed using a model pre-trained on the COCO dataset and training it on the UAVVwaste dataset to take advantage of transfer learning. The anchor boxes were generated using a procedure recommended since YOLOv2, which involves k-means clustering using intersection over union as a metric. This method adapts the anchor boxes' dimensions to the objects found on images in the target dataset. It is an important preparation step since the anchor boxes in COCO have different sizes than ones found in UAVVwaste.

The detected objects are marked on the map using the UAVs onboard sensors. To relate the GPS/GNSS coordinates to the coordinates of the objects detected in the image, one first needs to ensure proper camera calibration. Since the objects' position in the acquired image and the real world coordinates are established based on the geometric relationships, lens distortion must be removed, and the centre of the image plane needs to be properly positioned. Proven intrinsic camera parameter calibration is available in many open source libraries such as OpenCV [58]; however, the dedicated Kalibr library provides more features [59]. Calibration of a sample camera with the chessboard pattern using Kalibr is shown in Figure 5.

Figure 5. Calibration of camera using Kalibr. On the left—snapshot of the chessboard pattern with the characteristic corner points found. On the right—image from the calibrated camera, rectified using the computed calibration data.

The world coordinates are established using detections of objects of interest on acquired images under the assumption that the camera always faces vertically down. The schematic depiction of the setup is shown in Figure 6, and the summary of the symbols is given in Table 1.

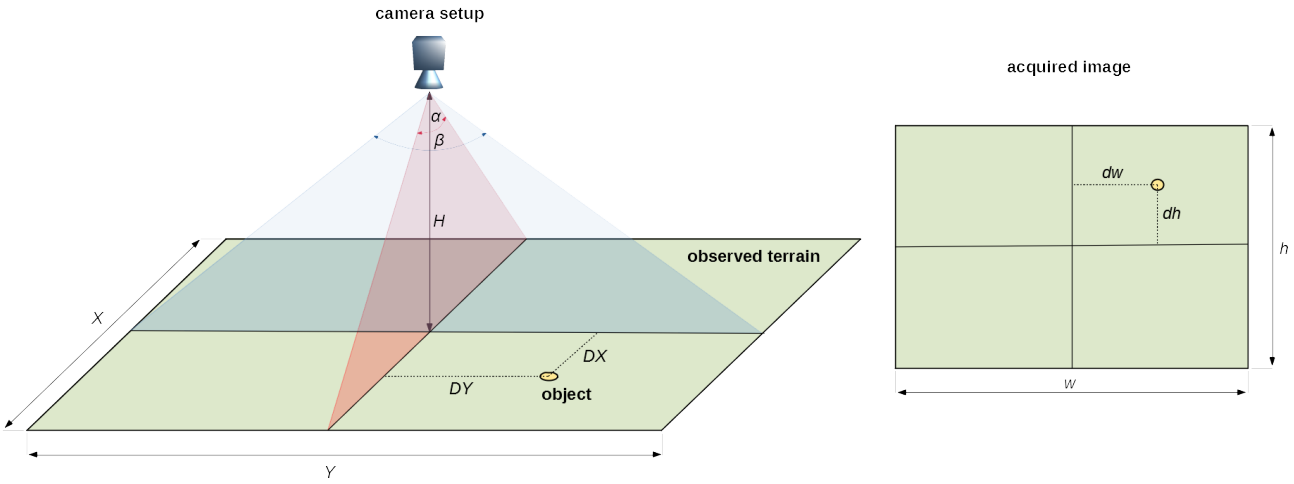

Figure 6. Schematic depiction of the camera setup and the captured image with characteristic measurements.

Table 1. Description of designations used in Figure 6 and in further equations.

Designation	Description
α	horizontal camera viewing angle [°]
β	vertical camera viewing angle [°]
H	UAV altitude relative to ground level [m]
X	width of the terrain area observed by the camera [m]
Y	height of the terrain area observed by the camera [m]
w	image width [px]
h	image height [px]
dx	displacement of object's b. box center relative to the image center along the x-axis [px]
dw	displacement of object's b. box center relative to the image center along the y-axis [px]
DX	displacement of object's center relative to the observed area center along the x-axis [m]
DY	displacement of object's center relative to the observed area center along the y-axis [m]

With the knowledge of the camera's viewing angles and the UAV's relative height, the size of the area observed by the camera can be computed as shown in Equation (1):

$$\begin{aligned} \tan(\frac{\alpha}{2}) &= \frac{\frac{1}{2}X}{H} \quad \rightarrow \quad X = 2H\tan(\frac{\alpha}{2}) \\ \tan(\frac{\beta}{2}) &= \frac{\frac{1}{2}Y}{H} \quad \rightarrow \quad Y = 2H\tan(\frac{\beta}{2}) \end{aligned} \tag{1}$$

Since the image is formed in the camera sensor as a consequence of projections, from the similarity of triangles, we can write Equation (2).

$$\begin{aligned} \frac{DX}{X} &= \frac{dw}{w} \\ \frac{DX}{Y} &= \frac{dh}{h} \end{aligned} \tag{2}$$

Substituting Equations (1) amd (2) after the correction by the UAV yaw angle measured by the IMU compass yields the final displacement of the detected object relative to the centre of the area underneath the UAV and observed by the camera, which coincides with the UAVs' coordinates that are available as measurements from the autopilot's GPS receiver:

$$\begin{aligned} DX &= 2H\frac{dw}{w}\tan(\frac{\alpha}{2}) \\ DY &= 2H\frac{dh}{h}\tan(\frac{\beta}{2}) \end{aligned} \tag{3}$$

The camera and image-related parameters can be established with relatively high accuracy, so the overall object localisation accuracy on the map depends mostly on the accuracy of altitude measurement, and, as seen in Equation (3), the overall error will be proportional to the error of altitude measurement. Since perfect localisation is not required (effective pickup is possible even if the localisation is off by a meter or more), a wide range of sensors (GPS, barometric, LiDAR, vision) with varying characteristics and accuracy may be used [60]. However, in case high object localisation precision is required (e.g., for automated pickup by autonomous vehicles), using a high precision altimeter is recommended. The localisation precision can be further increased by using more sophisticated navigation techniques such as simultaneous localisation and mapping or visual odometry, which have been successfully employed on UAVs [61,62].

3. Results

The accuracy was evaluated with selected protocols used in the COCO dataset detection task evaluation using the *cocoeval* software benchmark. The basic metric computed is the mean average precision (mAP) metric, but in a range of variants depending on the maximum number of objects detected, the object size and the threshold overlap computer as of image over union (IoU) upon which a successful detection is called. From the overall set of metrics generated by *cocoeval*, the following subset was selected as the most informative:

1. M1-mAP at a fixed IoU of 50%, also called mAP@50. Since such degree of overlap of predicted and ground truth bounding boxes for small objects is sufficient, and we are mostly interested in whether or not an object was detected to put it on the map. This is the main detection quality metric.
2. M2—For this metric, the detections are performed with IoU ranging from 0.5 to 0.95 with a 0.05 step. For each step, all detections with IoU in this range are considered positive detections, so, as the threshold IoU increases, there will be less true positive detections. All the detections across all steps contribute towards the final mAP computation by averaging their partial results. This is the main quality metric for the detection task in the COCO competition.

3. M3, M4, and M5 is a set of metrics computed as M2, but for small objects (M3—Longer bounding box side under 32 pixels), medium-size objects (M4—Longer bounding box side 32 to 96 pixels) and large objects (M5—longer bounding box side over 96 pixels).
4. M6, M7, and M8 are recall values directly corresponding to (and computed for the same sets of detections as) M3 to M5. These are included to give an overview of how many objects are missed and to give an idea of their breakdown by size.

The metrics were computed for all tested neural network architectures with the input images resized to 608×608 pixels. Moreover, the evaluation was performed individually for the full precision and half-precision floating-point format and the quantized 8-bit integer format whenever possible to give an overview of the numerical representation's impact on accuracy. The type conversion was performed using the TensorRT tool with the exception of the USB accelerators using their own dedicated compilers and converters. The results are collected in Table 2.

Table 2. Accuracy results for the tested models and their variants with a range of numerical representations. Explanations on the meaning of M1-M8 metrics are given in text. Asterisk (*) in a column means that the model could not be compiled for the given combination of computational platform and data representation type.

Metric	**YOLOv4**			**YOLOv3**			**YOLOv4-CSP**			**YOLOv4-tiny-3l**		
	FP32	**FP16**	**INT8**	**FP32**	**FP16**	**INT8**	**FP32**	**FP16**	**INT8**	**FP32**	**FP16**	**INT8**
M1	**0.785**	0.784	0.166	0.694	0.694	0.075	0.736	0.745		0.615	0.614	
M2	**0.476**	0.473	0.101	0.342	0.340	0.040	0.424	0.424		0.336	0.335	
M3	**0.282**	0.280	0.014	0.152	0.152	0.000	0.219	0.219		0.137	0.144	
M4	0.546	0.542	0.128	0.416	0.415	0.040	0.498	0.498	*	0.422	0.417	*
M5	0.611	0.603	0.332	0.373	0.373	0.274	0.561	0.559		0.335	0.335	
M6	0.330	0.334	0.011	0.218	0.216	0.000	0.276	0.275		0.218	0.226	
M7	0.593	0.590	0.137	0.472	0.472	0.046	0.558	0.558		0.472	0.470	
M8	0.639	0.635	0.352	0.400	0.400	0.304	0.591	0.591		0.361	0.361	

Metric	**YOLOv4-tiny**			**YOLOv3-tiny**			**EfficientDet-d1**		**EfficientDet-d3**		**MobileNetV2 SSD**		
	FP32	**FP16**	**INT8**	**FP32**	**FP16**	**INT8**	**FP32**	**FP16**	**FP32**	**FP16**	**FP32**	**FP16**	**INT8**
M1	0.566	0.566	0.270	0.239	0.232	0.000	0.669	0.669	0.751	0.750	0.545		0.000
M2	0.280	0.281	0.109	0.064	0.063	0.000	0.338	0.338	0.440	0.445	0.255		0.000
M3	0.131	0.132	0.074	0.018	0.018	0.000	0.097	0.095	0.150	0.153	0.068		0.000
M4	0.358	0.360	0.139	0.085	0.084	0.000	0.440	0.444	0.548	**0.555**	0.343		0.000
M5	0.114	0.114	0.000	0.023	0.023	0.000	0.589	0.589	0.612	**0.612**	0.409	*	0.000
M6	0.199	0.201	0.110	0.047	0.049	0.000	0.239	0.241	**0.350**	0.346	0.180		0.000
M7	0.411	0.412	0.169	0.140	0.139	0.000	0.549	0.556	0.633	**0.639**	0.450		0.000
M8	0.113	0.113	0.000	0.022	0.022	0.000	0.635	0.635	0.648	**0.652**	0.527		0.000

YOLOv4 full variant is a clear winner in terms of the main metric, with EfficientDet-d3 coming in second. Interestingly, EfficientDet-d3 outperforms YOLOv4 in terms of recall metrics in all relevant categories. Therefore, we can conclude that, while the former does not miss as many objects of interest, it also produces more false-positive detections. Interestingly, the CSP variant of YOLOv4 utilizing compound scaling turned out to perform worse than the basic version in terms of all metrics, which means that results (in terms of method ranking) reported for the COCO dataset must not be representative for any given dataset and might depend on the application domain. Previous generation YOLOv3 fall behind all the 'large' networks but EfficientDet-d1. It is also clearly inferior to other methods in terms of accuracy. Both 'tiny' variants of YOLOv4 performed surprisingly well

in terms of the M1 metric. The drop from the full to the reduced variant of the neural network is less prominent than it is the case with YOLOv3. Interestingly, the reduced YOLO variants seem to suffer the most in terms of reduced recall while detecting large objects. Other networks seem to behave in a more expected manner—detection of larger objects is easier than detecting small objects. The SSD detector with the lightweight MobileNet backend performs worse than the 'large' YOLO and EfficientDet networks. Still, it easily holds its ground versus the 'lite' variants, especially in terms of recall. Converting the numerical representation from full floating-point precision to FP16 does not seem to make the results noticeably worse. Interestingly, the results improve slightly in some cases. On the other hand, quantisation results in severe performance degradation, which is a clear indicator that straightforward quantization is not possible for most models and requires additional architectural considerations and low-level optimisations [63]. In some cases, the performance falls to zero or slightly above that, rendering the model useless.

Timing performance for inference on the GPU of the Nvidia Xavier NX platform with a range of models and data types are given in Table 3.

Table 3. Performance results for Nvidia Xavier NX for the tested models and their variants using a range of types for numerical representation. ATPI stands for average time per inference computed by dividing the overall batch processing time by batch size. The results for batch size yielding the maximum ATPI are displayed, as they correspond to maximum theoretical performance. Performance terms of images processed per second is also given. The asterisk (*) means that the model could not be compiled for the computational platform.

Neural Network Variant	ATPI [s]	FPS
YOLOv4 FP32	0.1868	5.352
YOLOv4 FP16	0.0519	19.262
YOLOv4 INT8	0.0288	34.709
YOLOv3 FP32	0.1883	5.317
YOLOv3 FP16	0.0475	21.054
YOLOv3 INT8	0.0235	42.513
YOLOv4-CSP FP32	0.1554	6.435
YOLOv4-CSP FP16	0.0429	23.308
YOLOv4-CSP INT8	*	
YOLOv4-tiny-3l FP32	0.0258	38.732
YOLOv4-tiny-3l FP16	0.0086	116.560
YOLOv4-tiny-3l INT8	*	
YOLOv4-tiny FP32	0.0223	44.889
YOLOv4-tiny FP16	0.0076	132.245
YOLOv4-tiny INT8	0.0057	176.0286
YOLOv3-tiny FP32	0.0195	51.158
YOLOv3-tiny FP16	0.0074	135.350
YOLOv3-tiny INT8	0.0045	222.589
EfficientDet-d1 FP32	0.1143	8.745
EfficientDet-d1 FP16	0.0609	16.415
EfficientDet-d3 FP32	0.3547	2.819
EfficientDet-d3 FP16	0.1805	5.540

The results clearly show that the high accuracy of more complex models comes at the cost of decreased inference speed. The larger EfficientDet-d3 with full floating-point

representation runs at a bit less than three frames per second, followed by all the full tested YOLO variants with the speeds ranging from 5.3 to 6.4 frames per second. Full EfficientDet-d1 reaches almost nine frames per second. Significant performance gains can be achieved by using half-precision floating-point for inference. This is because the platform can perform half-precision operations directly by design, and using half-precision saves memory, enabling larger efficient batch sizes. This facilitates reaching full parallel computational performance. Considering that such conversion does not hinder the accuracy, it is certainly a step worth taking when deploying the final version of the trained model. Overall, the models can perform inference with the speed that is sufficient for the described application. Half precision YOLOv4 seems to be the best choice in this case, with its relatively high speed and aforementioned good detection accuracy. In case even higher inference speed is necessary, one can use the lightweight, 'tiny' variants of the YOLO networks to reach well over 100 inferences per second with the half-precision floating-point variants. Integer variants of all the neural network models are even faster than half-precision floating-point variants. Nevertheless, the aforementioned impact of quantization on detection accuracy shows that more advanced, possibly dedicated conversion methods must be applied to consider using such models in practice.

Timing performance of inference using the Coral USB accelerator is presented in Table 4.

Table 4. Performance results for the Coral USB accelerator with two USB interface options and two performance settings. ATPI stands for average time per inference, and FPS is frames per second.

	Libedgetpu1-Std				**Libedgetpu1-Max**			
	USB 3.0		**USB 2.0**		**USB 3.0**		**USB 2.0**	
	ATPI [s]	**FPS**	**ATPI [s]**	**FPS**	**ATPI [s]**	**FPS**	**ATPI [s]**	**FPS**
SSD	0.0418	23.9266	0.0862	11.5996	0.0316	31.6766	0.0764	13.0832

As mentioned, the deployment of models on the Coral is limited, and it is recommended to use the solutions made available for applications by the producer. Such models are optimized for execution on the specific computational architecture of the accelerator. Some specific model parts (like the feature pyramid in new YOLO variants) are hard to map to specific data types and execution units without hindering the performance. This is why the tests performed for the Coral are based on the provided SSD detector implementation with MobileNetV2 backend. As shown in the table, the processing speed is respectable, especially considering the accelerator's low price and power consumption. Using USB3.0 is highly recommended, as USB2.0 clearly poses a bottleneck limiting the theoretical throughput with relatively slow data transmission. For additional speedup, one can enable the high-performance mode, resulting in the processing speed of over 30 frames per second in the presented test case. However, one has to keep in mind to assure proper cooling.

The NCS2 can run the full and tiny variants of YOLOv4 and YOLOv3 due to its capability to perform the half-precision floating-point operations. The performance of the implemented models using USB3.0 and USB2.0 connection is given in Table 5.

The full versions of YOLO reach the speed of around one frame per second on the faster USB3.0. The relative performance drop while using USB2.0 is less noticeable than with the Coral since less data are being transferred per time unit. Using the 'tiny' variants results in significant speedup. Their performance in terms of accuracy is on par with SSD as shown in Table 2, so they can be seen as a viable option for USB-based accelerators if the full variants become too slow.

For comparison, selected neural network models were also tested on the embedded CPUs housed on the Nvidia Xavier NX and Raspberry Pi 4 boards. The results are shown in Table 6.

Table 5. Performance result for the NCS2 accelerator with two USB interface options. ATPI stands for average time per inference, FPS is frames per second.

	USB 3.0		**USB 2.0**	
	ATPI [s]	**FPS**	**ATPI [s]**	**FPS**
YOLOv4	1.0273	0.9734	1.1863	0.8429
YOLOv4-tiny	0.0894	11.1885	0.1229	8.1388
YOLOv3	1.1000	0.9091	1.2471	0.8019
YOLOv3-tiny	0.2064	4.8453	0.2729	3.6642

Table 6. Performance result for the Nvidia Xavier NX and Raspberry Pi 4 CPUs. ATPI stands for average time per inference, FPS is frames per second.

Jetson Xavier NX—tf.lite CPU NVIDIA Carmel ARMv8.2	**ATPI [s]**	**FPS**
EfficientDet-d1	1.9070	0.524
EfficientDet-d3	6.9869	0.143
YOLOv4	16.981	0.059
YOLOv4-tiny	1.6994	0.588
SSD	0.5691	1.757
Raspberry Pi 4B—tf.lite CPU BCM271, Cortex-A72x4	**ATPI [s]**	**FPS**
EfficientDet-d1	3.6274	0.2756
EfficientDet-d3	11.8212	0.0846
YOLOv4	14.4206	0.0693
YOLOv4-tiny	1.0848	0.9218
SSD	0.8236	1.2141

It is clear that the larger networks are really complex for this computational platform—The serial CPUs are clearly not designed to cope with massively parallel workloads such as deep neural networks. Still, the SSD model is capable of reaching the speed of almost two frames per second. Nevertheless, dedicated platforms such as embedded GPUs or USB accelerators are clearly the superior choice for this task.

4. Discussion

Overall, the Nvidia Xavier NX platform delivers the best flexibility in terms of the range of solutions that it can run. It also provides headroom for potential future developments. Using it with the YOLOv4 as the object detection model enables near real-time performance with good accuracy. If low power and low cost are a priority, the popular Raspberry Pi computer with dedicated USB accelerators offers an interesting alternative. However, one has to consider the accuracy trade-offs if opting for such a solution.

Sample predictions performed by YOLOv4 on images from the UAVVaste dataset are shown in Figure 7.

Figure 7. Sample predictions performed using YOLOv4 on selected images from the UAVVaste dataset.

The figure highlights some of the interesting characteristics of the solution. For example, the detector reliably distinguishes the litter from the stones, even though they seem perceptually similar. The lower right image shows an additional, unwanted bounding box being predicted in the bottom part of the image. Please note that litter oftentimes appears in clusters. Detecting additional objects or missing detections in a cluster lowers the performance measured using standard object detection metrics. Still, it does not affect the solution's usefulness, since all the litter in a cluster will be cleaned up after pinpointing its location on the map. Assuming a single registered image covers a 8 × 6 m area, 10 m/s speed would result in a 1.5FPS minimum required frame rate with a slight overlap. This enough to cover well over 0.1 km^2 area during a 30 min mission. However, higher processing speed is desirable, since performing the detection multiple times and aggregating the results might improve the quality of results. Moreover, it would be required in the case of using one of the most effective approaches for boosting the small object detection performance—using high resolution input images divided into parts. For higher altitude flights, one has to keep in mind that the effective size of the objects to be detected in pixels needs to remain roughly the same in order to retain the detection accuracy. This means that, for high altitude flights, one would need to increase the resolution of the image sensor, divide the input image into portions, and perform multiple detection operations, one for each such image portion. The capability to process more images per second would therefore be desirable in both cases. With that in mind, the Xavier NX running FP16 version of YOLOv4 is the most reliable, expandable option.

One has to keep in mind that aerial monitoring at the level of precision and resolution necessary to detect objects of typical litter size can be seen as a threat to privacy and can raise ethical concerns. The presented solution is enabled through autonomous, on-board processing and is designed for a specific application, in which the input data in the form of images are translated directly to an object of interest position on the map. Nevertheless, when considering deployment of such devices, extreme caution must be exercised to ensure proper level of safety. The review of UAV-related privacy and safety aspects presented in [64] cites the communication channel and drone software hijacking as the most important threats and venues of possible attack, so additional measures, e.g., in the form of software audit, are advised as a safety measure against such potential sources of problems.

5. Conclusions

A solution for automated trash and litter detection based on low-altitude UAV imagery was presented in this paper. The system is composed of low cost, off-the-shelf components that are easily integrated into a complete solution that operates onboard the UAV carrying out the patrol mission. The algorithm's core is the deep neural network for object detection via a calibrated, onboard camera. Supported by additional autopilot sensor measurements (GPS/GNSS, altimeter, compass), the detected objects can be placed on the map with an accuracy that enables automated path planning for subsequent pickup. Due to the lack of data available for this specific domain, a dedicated dataset was collected and annotated. The dataset is made publicly available to lower the entry threshold and facilitate the research in the field. Since deep neural networks pose a significant computational load, a range of dedicated embedded devices enabling low power inference was tested in a wide range of configurations, so that an informed choice can be made when selecting the platform.

Future work plans include the extension of the dataset to provide more variety in the training data. New synthetic data generation by object pasting and blending into natural background images collected from low-altitude UAV flights is also planned. Integrating with services like OpenLitterMap is also considered, since the presented research seems to be a good fit and a natural extension to this project [65]. Furthermore, additional effort will be put into the development of detection methods that will be more tailored to this specific application area, mostly by adopting approaches that can more successfully cope with small object detection, e.g., utilising the high-resolution data without down-scaling by dividing the image into parts and performing multiple inferences for a single source image. Lower than expected accuracy of the YOLOv4 SCP variant might also be a result of relatively low input image resolution. Low recall rate for larger objects in some networks also suggests that automated methods for anchor point size estimation might not be the best when it comes to this specific application. Further research will be conducted to investigate this phenomenon.

Author Contributions: Conceptualization, M.K.; Data curation, M.P. and B.P.; Formal analysis, M.K.; Funding acquisition, K.W.; Investigation, M.K., M.P. and B.P.; Project administration, K.W.; Resources, K.W.; Software, M.P. and B.P.; Supervision, M.K.; Writing—original draft, M.K.; Writing—review & editing, M.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Polish Ministry of Science and Higher Education.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The UAVVaste dataset can be downloaded from: https://github.com/UAVVaste/UAVVaste (accessed on 1 March 2021).

Acknowledgments: The authors would like to thank Adam Bondyra and Dominik Pieczyński for their help with data collection.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Campbell, F. *People Who Litter*; ENCAMS: Wigan, UK, 2007.
2. Riccio, L.J. Management Science in New York's Department of Sanitation. *Interfaces* **1984**, *14*, 1–13. [CrossRef]
3. Dufour, C. Unpleasant or tedious jobs in the industrialised countries. *Int. Labour Rev.* **1978**, *117*, 405.
4. Proença, P.F.; Simões, P. TACO: Trash Annotations in Context for Litter Detection. *arXiv* **2020**, arXiv:2003.06975.
5. Lowe, D.G. Object recognition from local scale-invariant features. In Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 20–27 September 1999; Volume 2, pp. 1150–1157.
6. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005; Volume 1, pp. 886–893.
7. Felzenszwalb, P.F.; Girshick, R.B.; McAllester, D.; Ramanan, D. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *32*, 1627–1645. [CrossRef]

8. Malisiewicz, T.; Gupta, A.; Efros, A.A. Ensemble of exemplar-SVMs for object detection and beyond. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 89–96.
9. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 24–27 June 2014; pp. 580–587.
10. Girshick, R. Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
11. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *arXiv* **2015**, arXiv:1506.01497.
12. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
13. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 21–37.
14. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.
15. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2980–2988.
16. Elsken, T.; Metzen, J.H.; Hutter, F. Neural architecture search: A survey. *arXiv* **2018**, arXiv:1808.05377.
17. Real, E.; Aggarwal, A.; Huang, Y.; Le, Q.V. Regularized evolution for image classifier architecture search. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 21 January–1 February 2019; Volume 33, pp. 4780–4789.
18. Zoph, B.; Cubuk, E.D.; Ghiasi, G.; Lin, T.Y.; Shlens, J.; Le, Q.V. Learning data augmentation strategies for object detection. *arXiv* **2019**, arXiv:1906.11172.
19. Tan, M.; Pang, R.; Le, Q.V. Efficientdet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 10781–10790.
20. Tan, M.; Le, Q.V. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv* **2019**, arXiv:1905.11946.
21. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; pp. 740–755.
22. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934.
23. Misra, D. Mish: A self regularized non-monotonic neural activation function. *arXiv* **2019**, arXiv:1908.08681.
24. Ghiasi, G.; Lin, T.Y.; Le, Q.V. Dropblock: A regularization method for convolutional networks. *arXiv* **2018**, arXiv:1810.12890
25. Yang, Z.; Wang, Z.; Xu, W.; He, X.; Wang, Z.; Yin, Z. Region-aware Random Erasing. In Proceedings of the 2019 IEEE 19th International Conference on Communication Technology (ICCT), Xi'an, China, 16–19 October 2019; pp. 1699–1703.
26. Rezatofighi, H.; Tsoi, N.; Gwak, J.; Sadeghian, A.; Reid, I.; Savarese, S. Generalized intersection over union: A metric and a loss for bounding box regression. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 658–666.
27. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. Scaled-YOLOv4: Scaling Cross Stage Partial Network. *arXiv* **2020**, arXiv:2011.08036.
28. Säckinger, E.; Boser, B.E.; Bromley, J.M.; LeCun, Y.; Jackel, L.D. Application of the ANNA neural network chip to high-speed character recognition. *IEEE Trans. Neural Netw.* **1992**, *3*, 498–505. [CrossRef] [PubMed]
29. Jouppi, N.P.; Young, C.; Patil, N.; Patterson, D.; Agrawal, G.; Bajwa, R.; Bates, S.; Bhatia, S.; Boden, N.; Borchers, A.; others. In-datacenter performance analysis of a tensor processing unit. In Proceedings of the 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA), Toronto, ON, Canada, 24–28 June 2017; pp. 1–12.
30. Schneider, D. Deeper and cheaper machine learning [top tech 2017]. *IEEE Spectr.* **2017**, *54*, 42–43. [CrossRef]
31. Sugiarto, I.; Liu, G.; Davidson, S.; Plana, L.A.; Furber, S.B. High performance computing on spinnaker neuromorphic platform: A case study for energy efficient image processing. In Proceedings of the 2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC), Las Vegas, NV, USA, 9–11 December 2016; pp. 1–8.
32. Verhelst, M.; Moons, B. Embedded deep neural network processing: Algorithmic and processor techniques bring deep learning to IoT and edge devices. *IEEE Solid-State Circuits Mag.* **2017**, *9*, 55–65. [CrossRef]
33. Lin, D.; Talathi, S.; Annapureddy, S. Fixed point quantization of deep convolutional networks. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 2849–2858.
34. Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; Bengio, Y. Binarized neural networks. In Proceedings of the 30th International Conference on Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 4107–4115.
35. Cheng, J.; Wang, P.s.; Li, G.; Hu, Q.h.; Lu, H.q. Recent advances in efficient computation of deep convolutional neural networks. *Front. Inf. Technol. Electron. Eng.* **2018**, *19*, 64–77. [CrossRef]
36. Cheng, Y.; Wang, D.; Zhou, P.; Zhang, T. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Process. Mag.* **2018**, *35*, 126–136. [CrossRef]
37. Meier, L.; Tanskanen, P.; Heng, L.; Lee, G.H.; Fraundorfer, F.; Pollefeys, M. PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Auton. Robot.* **2012**, *33*, 21–39. [CrossRef]

38. Ebeid, E.; Skriver, M.; Jin, J. A survey on open-source flight control platforms of unmanned aerial vehicle. In Proceedings of the 2017 Euromicro Conference on Digital System Design (DSD), Vienna, Austria, 30 August–1 October 2017; pp. 396–402.
39. Franklin, D.; Hariharapura, S.S.; Todd, S. Bringing Cloud-Native Agility to Edge AI Devices with the NVIDIA Jetson Xavier NX Developer Kit. 2020. Available online: https://developer.nvidia.com/blog/bringing-cloud-native-agility-to-edge-ai-with-jetson-xavier-nx/ (accessed on 15 December 2012)
40. Upton, E.; Halfacree, G. *Raspberry Pi User Guide*; John Wiley & Sons: Hoboken, NJ, USA, 2014.
41. Libutti, L.A.; Igual, F.D.; Pinuel, L.; De Giusti, L.; Naiouf, M. Benchmarking performance and power of USB accelerators for inference with MLPerf. In Proceedings of the 2nd Workshop on Accelerated Machine Learning (AccML), Valencia, Spain, 31 May 2020.
42. Mittal, P.; Sharma, A.; Singh, R. Deep learning-based object detection in low-altitude UAV datasets: A survey. *Image Vis. Comput.* **2020**, *104*, 104046. [CrossRef]
43. Ammour, N.; Alhichri, H.; Bazi, Y.; Benjdira, B.; Alajlan, N.; Zuair, M. Deep learning approach for car detection in UAV imagery. *Remote. Sens.* **2017**, *9*, 312. [CrossRef]
44. Wang, X.; Cheng, P.; Liu, X.; Uzochukwu, B. Fast and accurate, convolutional neural network based approach for object detection from UAV. In Proceedings of the IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society, Washington, DC, USA, 21–23 October 2018; pp. 3171–3175.
45. Zhang, X.; Izquierdo, E.; Chandramouli, K. Dense and small object detection in uav vision based on cascade network. In Proceedings of the IEEE International Conference on Computer Vision Workshops, Seoul, Korea, 27 October–2 November 2019.
46. Tong, K.; Wu, Y.; Zhou, F. Recent advances in small object detection based on deep learning: A review. *Image Vis. Comput.* **2020**, *97*, 103910. [CrossRef]
47. Robicquet, A.; Sadeghian, A.; Alahi, A.; Savarese, S. Learning social etiquette: Human trajectory understanding in crowded scenes. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016; pp. 549–565.
48. Zhu, P.; Wen, L.; Du, D.; Bian, X.; Hu, Q.; Ling, H. Vision Meets Drones: Past, Present and Future. *arXiv* **2020**, arXiv:2001.06303.
49. Božić-Štulić, D.; Marušić, Ž.; Gotovac, S. Deep learning approach in aerial imagery for supporting land search and rescue missions. *Int. J. Comput. Vis.* **2019**, *127*, 1256–1278. [CrossRef]
50. Lo, H.S.; Wong, L.C.; Kwok, S.H.; Lee, Y.K.; Po, B.H.K.; Wong, C.Y.; Tam, N.F.Y.; Cheung, S.G. Field test of beach litter assessment by commercial aerial drone. *Mar. Pollut. Bull.* **2020**, *151*, 110823. [CrossRef] [PubMed]
51. Merlino, S.; Paterni, M.; Berton, A.; Massetti, L. Unmanned Aerial Vehicles for Debris Survey in Coastal Areas: Long-Term Monitoring Programme to Study Spatial and Temporal Accumulation of the Dynamics of Beached Marine Litter. *Remote. Sens.* **2020**, *12*, 1260. [CrossRef]
52. Nazerdeylami, A.; Majidi, B.; Movaghar, A. Autonomous litter surveying and human activity monitoring for governance intelligence in coastal eco-cyber-physical systems. *Ocean. Coast. Manag.* **2021**, *200*, 105478. [CrossRef]
53. Hong, J.; Fulton, M.; Sattar, J. A Generative Approach Towards Improved Robotic Detection of Marine Litter. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 10525–10531.
54. Panwar, H.; Gupta, P.; Siddiqui, M.K.; Morales-Menendez, R.; Bhardwaj, P.; Sharma, S.; Sarker, I.H. AquaVision: Automating the detection of waste in water bodies using deep transfer learning. *Case Stud. Chem. Environ. Eng.* **2020**, *2*, 100026. [CrossRef]
55. Gorbachev, Y.; Fedorov, M.; Slavutin, I.; Tugarev, A.; Fatekhov, M.; Tarkan, Y. OpenVINO deep learning workbench: Comprehensive analysis and tuning of neural networks inference. In Proceedings of the IEEE International Conference on Computer Vision Workshops, Seoul, Korea, 27–28 October 2019.
56. Lee, J.; Chirkov, N.; Ignasheva, E.; Pisarchyk, Y.; Shieh, M.; Riccardi, F.; Sarokin, R.; Kulik, A.; Grundmann, M. On-device neural net inference with mobile gpus. *arXiv* **2019**, arXiv:1907.01989.
57. Gray, A.; Gottbrath, C.; Olson, R.; Prasanna, S. Deploying Deep Neural Networks with NVIDIA TensorRT. 2017. Available online: https://developer.nvidia.com/blog/deploying-deep-learning-nvidia-tensorrt/ (accessed on 15 December 2012).
58. Kaehler, A.; Bradski, G. *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*; O'Reilly Media, Inc.: Newton, MA, USA, 2016.
59. Rehder, J.; Nikolic, J.; Schneider, T.; Hinzmann, T.; Siegwart, R. Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 4304–4311.
60. Geng, K.; Chulin, N. Applications of multi-height sensors data fusion and fault-tolerant Kalman filter in integrated navigation system of UAV. *Procedia Comput. Sci.* **2017**, *103*, 231–238. [CrossRef]
61. Kanellakis, C.; Nikolakopoulos, G. Survey on computer vision for UAVs: Current developments and trends. *J. Intell. Robot. Syst.* **2017**, *87*, 141–168. [CrossRef]
62. Al-Kaff, A.; Martin, D.; Garcia, F.; de la Escalera, A.; Armingol, J.M. Survey of computer vision algorithms and applications for unmanned aerial vehicles. *Expert Syst. Appl.* **2018**, *92*, 447–463. [CrossRef]
63. Krishnamoorthi, R. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv* **2018**, arXiv:1806.08342.

64. Altawy, R.; Youssef, A.M. Security, privacy, and safety aspects of civilian drones: A survey. *ACM Trans. Cyber-Phys. Syst.* **2016**, *1*, 7. [CrossRef]
65. Lynch, S. OpenLitterMap. com–open data on plastic pollution with blockchain rewards (littercoin). *Open Geospat. Data Softw. Stand.* **2018**, *3*, 6. [CrossRef]