electronics

MDPI

Article

Towards Fully Autonomous UAV: Damaged Building-Opening Detection for Outdoor-Indoor Transition in Urban Search and Rescue

Ali Surojaya, Ning Zhang, John Ray Bergado * and Francesco Nex

Faculty of Geo-Information Science and Earth Observation (ITC), University of Twente, 7522 NH Enschede, The Netherlands; alisurojaya@student.utwente.nl (A.S.); n.zhang@utwente.nl (N.Z.); f.nex@utwente.nl (F.N.)
* Correspondence: j.r.bergado@utwente.nl

Abstract: Autonomous unmanned aerial vehicle (UAV) technology is a promising technology for minimizing human involvement in dangerous activities like urban search and rescue missions (USAR), both in indoor and outdoor. Automated navigation from outdoor to indoor environments is not trivial, as it encompasses the ability of a UAV to automatically map and locate the openings in a damaged building. This study focuses on developing a deep learning model for the detection of damaged building openings in real time. A novel damaged building-opening dataset containing images and mask annotations, as well as a comparison between single and multi-task learning-based detectors are given. The deep learning-based detector used in this study is based on YOLOv5. First, this study compared the different versions of YOLOv5 (i.e., small, medium, and large) capacity to perform damaged building-opening detections. Second, a multitask learning YOLOv5 was trained on the same dataset and compared with the single-task detector. The multitask learning (MTL) was developed based on the YOLOv5 object detection architecture, adding a segmentation branch jointly with the detection head. This study found that the MTL-based YOLOv5 can improve detection performance by combining detection and segmentation losses. The YOLOv5s-MTL trained on the damaged building-opening dataset obtained 0.648 mAP, an increase of 0.167 from the single-task-based network, while its inference speed was 73 frames per second on the tested platform.

Keywords: damaged building opening; multitask learning; YOLOv5; object detection; image segmentation

check for updates

Citation: Surojaya, A.; Zhang, N.; Bergado, J.R.; Nex, F. Towards Fully Autonomous UAV: Damaged Building-Opening Detection for Outdoor-Indoor Transition in Urban Search and Rescue. *Electronics* **2024**, *13*, 558. https://doi.org/10.3390/electronics13030558

Academic Editors: Yunda Yan, Dewei Yi, Hao Lu and Lan Gao

Received: 27 December 2023
Revised: 23 January 2024
Accepted: 26 January 2024
Published: 30 January 2024

1. Introduction

Search and rescue is one of the first, important and time-critical tasks of an emergency response [1]. In urban disasters like earthquakes, it aims to save people trapped in rubble and isolated due to collapsed buildings. However, most routines in urban search and rescue (USAR), such as inspecting affected buildings and continuing to search for victims, are still done manually by the disaster response team. This effort is not only time consuming and less effective, but also dangerous for first responders (FR) obliged to enter unknown and risky environments. For this reason, it is necessary to apply relevant technologies for post-disaster scenarios [2], limiting human involvement and reducing the risks within a search and rescue mission.

In this regard, the use of automated devices such as unmanned aerial vehicles (UAVs), is often adopted to limit the involvement of FR in USAR operations [3–5]. The development of semi-autonomous UAV systems has been already widely explored as a disaster management and response tool [1,6,7].

Most autonomous UAVs are normally operated exclusively outdoors, while their use inside buildings is less common. In any case, the capability to seamlessly navigate from an outdoor environment to the interior of a building is largely unexplored. Besides localization problems (i.e., moving from a GNSS to GNSS-denied environment), the first challenge in entering a collapsed building lies in the identification of a building opening (e.g., open doors). In a disaster scenario, this task is further complicated by partially collapsed elements and the presence of "unconventional" openings in the scene (e.g., holes in facades). A solution able to reliably map the openings in a damaged building is therefore needed to allow UAVs to enter without the need to be manually teleoperated by a pilot.

This paper aims to shed some light on this problem by proposing an automated opening detection algorithm to guide the seamless entrance of UAVs into damaged buildings. In this context, openings are damaged walls and holes, in addition to conventional opened windows and doors. If sufficiently large, all these kinds of openings can be a UAV access point into a building. To the best of our knowledge, no deep learning model has been developed specifically for this task and, in addition, no real-time implementations are available to allow UAV navigation. Specifically, in this study a real-time object detection algorithm was implemented to detect potential entrances on images of buildings affected by disasters, using a dedicated convolutional neural network (CNN). In particular, a multitask learning-based network was implemented by combining object detection and segmentation tasks. The underlying idea was to utilize the segmentation task to improve detection performance by classifying the pixels of an opening space. Such multiobjective optimization effectively enhances the robustness of the features learned by the network [8], and thus often help improve performance on the tasks being jointly learned. This algorithm was compared to traditional object detection algorithms to assess the performance in realistic conditions.

The main contributions of this research are (1) the preparation and annotation of a collapsed building opening dataset; (2) the development of a real-time object detection model to detect openings on images of disaster-affected buildings using state-of-the-art object detection; and (3) an innovative algorithm combining object detection and segmentation based on multitask learning networks to detect openings in damaged buildings. The datasets and the pretrained models are available for the scientific community, as it is the authors' belief that the results of this study could be used in different search and rescue missions.

2. Related Work

UAV has been widely applied to replace human involvement in urban search and rescue (USAR) [9–11]. The capability of an autonomous UAV to enter a building for a USAR mission is crucial to exploring an environment, giving useful information to prevent risks for rescuers. However, the identification of the openings of a collapsed building is not always an easy task. Thus, an automated solution to map the openings in a damaged building is necessary. UAVs can be equipped with various sensors and detection approaches to navigate the system in the process of entering a damaged building. The sections below briefly discuss some related works, dividing them into two categories according to the sensors used in navigation (i.e., active and passive sensors).

2.1. UAV with Active Sensors for Building Opening Detection

Active sensors, such as light detection and ranging (LiDAR), are commonly used for UAV navigation [12,13]. LiDAR measurements facilitate UAV navigation by producing high frequency laser pulses to measure distances, and generating precise three-dimensional maps of a surrounding environment [14]. This capacity to swiftly capture detailed topographical data allows UAVs to navigate through complex terrains, avoid obstacles, and execute intricate missions with remarkable precision.

The integration of global navigation satellite systems (GNSS), inertial navigation system (INS), and LiDAR was developed to facilitate accurate UAV navigation [15]. This technology is shown to be able to improve navigation performance by reducing RMSE for horizontal and vertical directions. LiDAR point clouds enable navigation systems to detect and avoid obstacles for UAVs [13]. In indoor environments scenarios, LiDAR sensors were utilized by Petrlik et al. [16] for UAVs stabilization, navigation, and localization. LiDAR localization shows robust performance in poorly illuminated indoor environments.

The combination of a depth camera, LiDAR, and a priori information [17] has been the state-of-the-art for autonomous UAVs outdoor–indoor transition. This technology is applied in firefighting scenarios to enable UAVs to recognize and estimate a window's location, orientation, and edges to enter a building autonomously. Their approach needs at least one combination of depth camera plus LiDAR, or LiDAR with a priori information, to support the automatic navigation for UAVs. In some cases, a UAV equipped with a depth camera and LiDAR is unavailable due to the cost and size of the needed platform. Thus, the possibility of only using an RGB camera (passive sensor) for UAV navigation will be studied in this research.

2.2. UAV with Passive Sensors for Building Opening Detection

Passive sensors such as cameras are widely used in UAV technology. Several approaches have already been developed to use cameras as the main information source for autonomous UAV navigation [18–20]. Most of these approaches implement deep learning algorithms mimicking human vision [21], such as object detection tasks. The main idea of the object detection algorithm is to develop a computational model to present information about where and what objects are [22]. An object detection model needs to be able to define the location of an object, and which category the object falls into (classification). Thus, an object detection workflow can be divided into three main stages, i.e., region selection, feature extraction, and object classification [23].

Previous work in opening detection presented three stages to detect and classify door openings using RGB and depth information [24]. That algorithm was used to classify a door into different opening classes, i.e., open, semi-open, and closed, successfully classifying doors in real time on a limited powered device used for indoor navigation purposes to navigate between rooms. The network developed in [24] was based on a sequential method that combined object detection with a classification network to detect and segment doors. The object detection allowed the cropping of parts of the initial images, that were then used as input to classify open, semi-open, or closed doors. The BiSeNet [25] was used to segment the door, and AlexNet to classify the open status of the doors. Despite good results, this approach was not additionally trained to detect door openings in damaged building scenarios, showing poor performance in the unideal condition of doors in damaged building scenarios. Also, compared to the current state of the art in real-time object detection networks, the inference time of the door detection and classification network is significantly higher because of the sequential approach.

Besides this work, no single study which has developed real-time opening detection in a damaged building scenario exists, although there have been several works of research in the field of autonomous UAVs for search and rescue missions. The work presented here tries to overcome the above-mentioned problems by proposing a detection algorithm specifically conceived and trained for detecting opening spaces in images of damaged buildings, working in real-time on low-powered devices.

3. Methodology

This study proposes a combination of object detection and image segmentation in a multitask learning setup, to improve the performance of a real-time damaged building-opening detection model combining convolutional neural networks for object detection and image segmentation. The idea was to use the segmentation task to enhance the performance of the detection model. Hard parameter sharing among tasks was employed in improving the architecture of the damaged building-opening detection model. This means that the multitask model consisted of a shared backbone followed by two task-specific heads, i.e., detection and segmentation heads. This sharing method is beneficial for regularization and reduces the risk of overfitting [26], especially with relatively small datasets.

While the object detection task tries to define the presence of objects and its position [22] in the image, image segmentation deals with pixel-based classification, dividing the scene into multiple segments [27,28]. For this study, YOLOv5 [29] was adopted as the starting point for the multitask network implementation. You only look once (YOLO) [30] is a well-known one-stage detector network that can perform real-time object detection. Although more recent implementations of this algorithm were available, the authors opted to use v5, as the results in terms of accuracy and processing time were extremely close, and the implementation of the multitask approach was easier to implement. To assess the improvement given by a multitask approach, two types of damaged building opening detection algorithms were compared in this study: the single-task object detection, and multitask learning model, described in the following sections.

3.1. YOLOv5 Single-Task Object Detection

Conventional object detection networks are based on a single-task approach. Thus, it was necessary also to train the single-task model to have a benchmark comparison for the proposed model approach (multitask learning). YOLOv5 is a one-stage detector that is currently one of the well-performing real-time object detection models. This network is pretrained on the COCO dataset with a fastest detection speed of up to 140 frames per second [31]. This model architecture consists of three main layer blocks: the backbone, neck, and head, as shown in Figure 1.

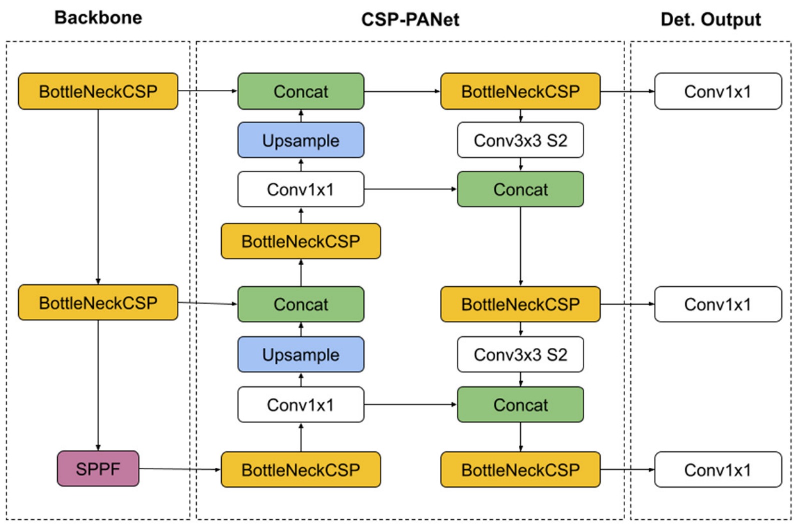

Figure 1. YOLOv5 version 6.0/6.1 model architecture consists of backbone, neck, and head CSP-PANet [29].

The backbone used in YOLOv5 version 6.0/6.1 is a new CSP-Darknet53, which is based on the cross stage partial network (CSPNet) [32] and Darknet53 [30]. The combination of CSP-Darknet53 is performed by reducing and replacing the convolutional residual blocks of Darknet53 with BottleneckCSP modules. This backbone is shown as the optimal network for a detector by providing a large receptive field size and parameter number [33].

YOLOv5 uses SPPF and the new CSP-PAN as a neck to combine different scales of features extracted from different levels of the backbone. SPPF is a modified version of spatial pyramid pooling or SPP [34], with fewer floating-point operations (FLOPs). The CSP-PAN network is used to aggregate the features. The detection head produces object detections from these multi-level aggregated features. Such extraction of multi-level features are usually done to effectively have large receptive fields in the backbone network [35], which is crucial for feature extraction in vision tasks dealing with high resolution images. The complete overview of YOLOv5 architecture is shown in Figure 1, and a corresponding pseudocode is presented in Algorithm 1.

Algorithm 1: Single-task Learning Based Detector

```
   Input: $I$ = Images of damaged buildings
   $f_{b(\omega)}$ = network backbone
   $f_{n(\omega)}$ = network neck
   $f_{d(\omega)}$ = network detection head
   Output: $B$ = Bounding boxes of detected openings
   $C$ = Confidence scores of the detected openings
1  Load model weights $\omega$
2  Run forward pass $f_{d(\omega)}(f_{n(\omega)}\left(f_{b(\omega)}(I)\right))$
3      Extract features using network backbone $H_\sigma = f_{b(\omega)}(I)$ at different scales $\sigma$
4      Aggregate the extracted features using network neck $H = f_{n(\omega)}(H_\sigma)$
5      Extract bounding boxes and confidence scores using detection head $B, C = f_{d(\omega)}(H)$
```

3.2. YOLOv5 Multitask Learning

A multitask learning-based object detector, called YOLOv5-MTL, was developed based on the YOLOv5 object detection architecture, by adding a segmentation branch jointly with the detection head. The architecture of YOLOv5-MTL is based on hard parameter sharing by utilizing one backbone with two heads (detection and segmentation). Multitask learning (MTL) is a machine learning paradigm that trains a model to perform multiple tasks simultaneously [36]. MTL can potentially improve the network's performance for related tasks by sharing complementary features and acting as a regularizer for each task at hand [37].

The hard parameter-sharing technique can also benefit the model by reducing the risk of overfitting [26]. Hard parameter sharing in the MTL network divides the parameter into shared and task-specific parameters. These task-specific parameters are optimized separately for each task but use a cumulative loss.

The main architecture of YOLOv5-MTL is based on the YOLOv5 detection network with added functionality for image segmentation. The idea of adding the segmentation branch to the last output from the CSP-PANet, as shown in Figure 2 and in the pseudocode described in Algorithm 2, is inspired by Mask R-CNN [38], which is proven as the optimal multitask architecture for object detection and image segmentation.

Algorithm 2: Multitask Learning-Based Detector

```
    Input: $I$ = Images of damaged buildings
    $f_{b(\omega)}$ = network backbone
    $f_{n(\omega)}$ = network neck
    $f_{d(\omega)}$ = network detection head
    $f_{s(\omega)}$ = network segmentation head
    Output: $B$ = Bounding boxes of detected openings
    $C$ = Confidence scores of the detected openings
    $M$ = Segmentation masks of detected openings
1   Load model weights $\omega$
2   Run forward pass $f_{d(\omega)}(f_{n(\omega)}\left(f_{b(\omega)}(I)\right)),\ f_{s(\omega)}(f_{n(\omega)}\left(f_{b(\omega)}(I)\right))$
3       Extract features using network backbone $H_\sigma = f_{b(\omega)}(I)$ at different scales $\sigma$
4       Aggregate the extracted features using network neck $H = f_{n(\omega)}(H_\sigma)$
5a      Extract bounding boxes and confidence scores using detection head $B, C = f_{d(\omega)}(H)$
5b      Extract segmentation masks using segmentation head $M = f_{s(\omega)}(H)$
```

The backbone used in this network is the same as in YOLOv5, which is the new CSP-DarkNet53. This backbone has shown good performance in the features extraction process to provide information for the object detection algorithm. The extracted features from different levels of backbones were combined using SPPF and New CSP-PAN as the Neck network. YOLOv5-MTL was then modified by adding the ability to do instance segmentation. In particular, ProtoNet [39], a fully connected neural network, was added along with the YOLOv5 object detection head. ProtoNet is constructed by a three-layer network of 2D convolutions with sigmoid linear unit (SiLU) [40] activation functions to produce prototype masks. This prototype generation branch was attached to a backbone layer to predict a set of prototype masks for the whole image scene, and to define the final instance segmentation head along with the final 2D convolutional layers.

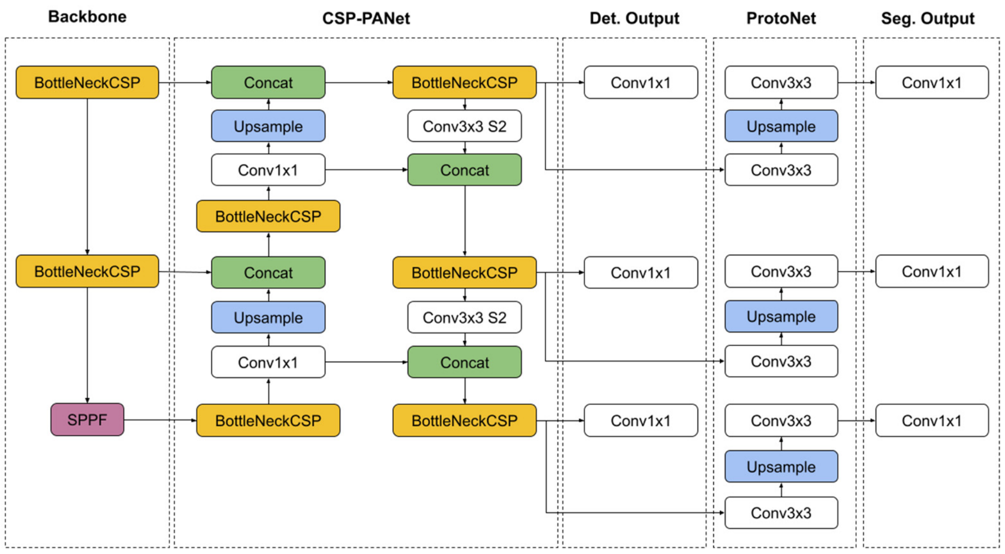

Figure 2. YOLOv5-MTL model architecture consists of backbone, neck, and detection head CSP-PANet [32], plus ProtoNet [39] for segmentation head.

As can be seen in Figure 2, ProtoNet is linked to deeper backbone features and can produce better robustness of the mask, and good performance on smaller objects [39]. This design can benefit a damaged building opening network to detect small openings.

The idea of developing a multitask learning-based detection model in this study is to improve the performance of the detection task by simultaneously performing a segmentation task. The losses of each task were combined to represent the model loss. The combined loss was calculated based on objectness loss, class probability loss, bounding box regression loss, and mask loss. The object and bounding box losses were responsible for showing the bounding box score prediction and coordinates, respectively. The class probability loss was used to obtain the object classification score, and the mask loss was responsible for the segmentation mask prediction. Regression loss for bounding box coordinates' prediction was calculated based on a mean intersection over union (IoU) ranging from 0–1. Object and class losses were calculated using the binary cross-entropy with the logits (BCEWithLogits) function available in most deep learning software libraries [41]. The BCEWithLogits loss function can be described as follows:

$$l(x,y) = L = \{l_1, \ldots, l_N\}^T,\ l_n = -\omega_n[y_n \cdot \log \sigma(x_n) + (1 - y_n) \cdot \log(1 - \sigma(x_n))], \tag{1}$$

where x is the input, y is the target, ω is the weight, N represents the batch size, and n is the sample number in the batch. BCEWithLogits Loss combines a sigmoid layer with BCE loss in one class to obtain more numerical stability than separating the sigmoid layer. This loss function resulted in a class probability prediction ranging from 0 to 1. Based on this prediction, the model will calculate the difference between the prediction and the ground truth, resulting in a loss value.

4. Dataset

This study used the RGB image dataset containing a scene of a damaged building obtained from previous work by [42] as the primary data to train the object detection model. The dataset contains images of earthquake events in Indonesia and Nepal with different scenes, such as damaged buildings, roads, bridges, and other collapsed structures. The 2579 images provided in this dataset were acquired from crowdsourcing through Twitter and Getty Images platforms. All images from Twitter are subject to the policy of free use for non-commercial usage, and the images from Getty Images are only used for training purposes. Most of these images were collected from terrestrial perspectives that are sufficiently similar to UAV data when flying at low altitude, capturing images at an oblique or near-horizontal angles, and in proximity of damaged buildings.

Not all images in this dataset [42] could be used for this study because we only focused on openings in damaged buildings. Thus, images with scenes of openings in damaged buildings needed to be selected. A total of 738 images were selected and divided into training, validation, and testing datasets, with a composition 70%, 20%, and 10%, respectively. The size of the images was 640 × 407 pixels; these were then converted into 640 × 640 pixels to obtain the square dimension of input images needed for model training purposes. The size conversion was done without affecting the aspect ratio by implementing image fits with black edges. This method resizes the image to the desired dimension without changing the image's aspect ratio, by filling the empty pixels with black color.

As part of the contribution of this study, the damaged building-opening dataset was developed from an available open-source image dataset. The dataset development consisted of image collection, selection, and annotation processes. Figure 3 shows some example images and their corresponding opening labels (in red polygon masks). The annotation masks and corresponding annotated images have been made publicly available by the authors (see our Data Availability Statement).

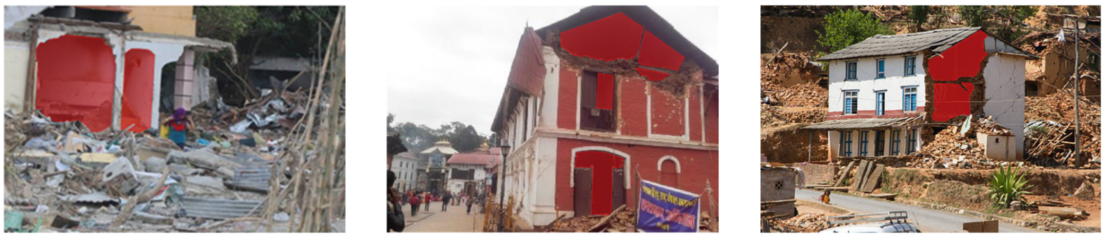

Figure 3. Damaged building images and the opening labels.

5. Results

This study was comprised of four main phases: firstly, a dataset was developed for detecting damaged building openings by curating and annotating images from an open-source dataset. Next, a single-task detection network was used to evaluate its ability in identifying these openings. Then, the multitask learning network was tested by integrating a segmentation task into the detection network architecture, aiming to enhance detection performance. Finally, the performance of both the single-task and multitask detection networks were compared in terms of accuracy and speed, utilizing YOLOv5 as the foundational network for this study.

Single and multitask learning object detection models were trained with the same settings to obtain comparable detection model performances. The hyperparameter values were chosen based on the standard YOLOv5 detection model training method. The model training specifications included a batch size of 16 and were run through 100 epochs. For optimization, stochastic gradient descent (SGD) was used. The chosen loss function was binary cross-entropy with logits loss, with the learning rate set at 0.01. This training was conducted on a Google Colab instance equipped with a 12 GB NVIDIA Tesla T4 GPU.

The trained network was evaluated by calculating several metrics on the test dataset. These evaluation metrics are measured to obtain the model's performance scores in performing the object detection or image segmentation task. Precision measures the model's performance in obtaining true positives from all positive predictions. In comparison, recall measures how good the model is at obtaining true positives from all the ground truth positives. The mAP is the average precision over classes. This average precision (AP) is given by the area under the precision-recall curve for the detectors [43]. The Precision-recall (PR) curve is a plotted graph that shows the recall change for a given precision and vice versa.

In the object detection domain, intersection over union (IoU) is a metric to measure the amount of predicted bounding box that overlaps with the on-the-ground reality bounding box, divided by the total area of both bounding boxes. This IoU was used to calculate the mAP as a threshold value. For instance, an IoU of 0.5 was used in mAP50 (i.e., the detection was classified as true positive if it had an IoU above 0.5). This study used mAP50 to evaluate the performance of the damaged building opening detection mode.

Three versions of YOLOv5 were trained on the dataset to provide better options for the damaged building-opening detection model. Small, medium, and large versions of YOLOv5 were trained and evaluated on the damaged building-opening dataset. This study aimed to examine how the size of the architecture, determined by various network hyperparameters, affects the inference speed which is crucial for real-time applications.

5.1. Opening Detection with YOLOv5

The YOLOv5 models trained on the damaged building-opening dataset were evaluated on test data which were not used in the training process. The models were tested on the Google Colab platform with NVIDIA Tesla T4 GPU. Table 1 shows the models' performance evaluated by calculating precision, recall, mean average precision (mAP), and inference speed in frames per second.

Table 1. Performance comparison of YOLOv5 detection models trained on the damaged building-opening dataset.

Model	Precision	Recall	mAP50	Speed (FPS)
YOLOv5s	0.609	**0.477**	0.481	**107**
YOLOv5m	**0.656**	0.450	**0.501**	52
YOLOv5l	0.616	0.455	0.490	32

The small, medium, and large versions of YOLOv5 showed slightly similar performance results in terms of precision, recall, and mAP. YOLOv5s had a smaller mAP compared to the other two but had the fastest inference speed. This result showed that the small version of the model was more efficient in terms of processing time. Interestingly, YOLOv5s had the highest recall but lowest precision among the three YOLOv5 models, indicating a slight increase in the number of true positives in exchange for detecting more false positive openings. Figure 4 shows some examples of detection results from YOLOv5s on the test images. From the visual interpretation of images in Figure 4, we can see that this model successfully detected several types of openings. Open doors, windows, and collapsed walls were detected in the scene. This result shows that the YOLOv5 model can detect damaged building openings by providing relevant training data.

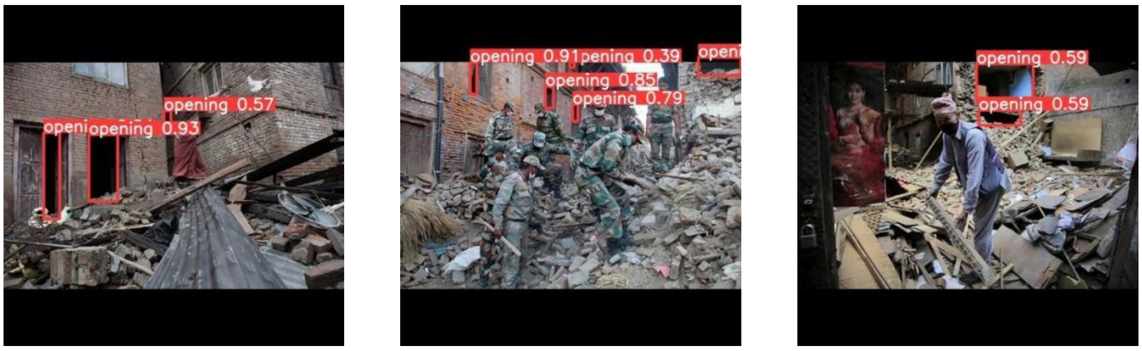

Figure 4. YOLOv5s detection output on the test dataset. Openings such as open doors, windows, and collapsed walls were successfully detected.

However, inference results on the test dataset also showed several detection errors. On the left images in Figure 5, some openings are not detected as an opening (false negative). Furthermore, some nonbuilding openings were detected as openings (false positive), as shown in the middle and right images in Figure 5. Similar visual characteristics of some non-opening and real opening objects led to this wrong detection result. The main challenge in detecting damaged building openings is the complexity of the opening's form, size, and shape. As there are no unique characteristics in the size and form of the openings, the opening pattern could be detected by better understanding the spatial context relative to other objects like walls and related objects like windows and doors.

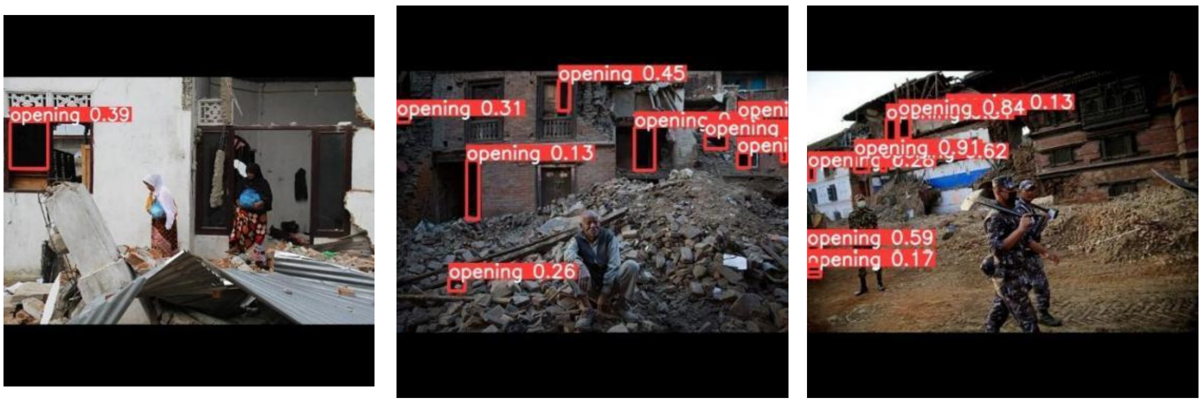

Figure 5. YOLOv5s detection output on the test dataset. False negatives and false positives were detected.

In that regard, a better understanding of the neighborhood scene would be needed, and image segmentation could provide support in contextualizing a larger part of the scene [44]. The idea of multitask learning by combining object detection and segmentation was therefore proposed to obtain better damaged building-opening detection.

5.2. Opening Detection with YOLOv5 MTL

After knowing the performance of the damaged building-opening detection model trained on YOLOv5 single-task object detection networks, this study examined the application of YOLOv5-MTL for this detection purpose. The multitask learning version of YOLOv5 was trained using the same training validation test dataset and training configuration as in YOLOv5 single-task, and was evaluated using the same metrics to evaluate the detection and segmentation performance.

Table 2 shows the performances of the YOLOv5-MTL with three different architecture versions. The large version of YOLOv5-MTL showed better mAP for detection and segmentation than the small and medium models. In terms of inference speed, the small version of the model was more than two times faster than the medium and large versions. This condition shows that the larger model size will result in more model parameters and more time for image inference.

Table 2. Performance comparison of YOLOv5 MTL models trained on damaged building-opening dataset.

Model	Detection			Segmentation			Speed (FPS)
	P	R	mAP50	P	R	mAP50	
YOLOv5s-MTL	0.747	0.577	0.648	0.675	0.539	0.585	**73**
YOLOv5m-MTL	0.753	0.606	0.664	0.671	0.540	0.577	30
YOLOv5l-MTL	**0.799**	**0.629**	**0.698**	**0.732**	**0.556**	**0.611**	27

YOLOv5l-MTL was not able to perform real-time inference (<30FPS), while the medium version barely met the real-time requirement. The complexity of the model influenced these inference speed performances. The difference between the mAP of the small and medium models was only about 0.02. Thus, considering the inference speed, the YOLOv5s-MTL model could be considered the most balanced option for a damaged building-opening detection model. Unlike the single-task detection networks, the YOLOv5s-MTL provided both bounding boxes and segmentation masks in correspondence with openings. The prediction output provided information about the class name, class probability, bounding box, and masks, as shown in Figure 6. As in the detection output of a single-task detector (Figure 4), the MTL version of YOLOv5 can correctly detect the openings with additional information, i.e., a segmentation mask.

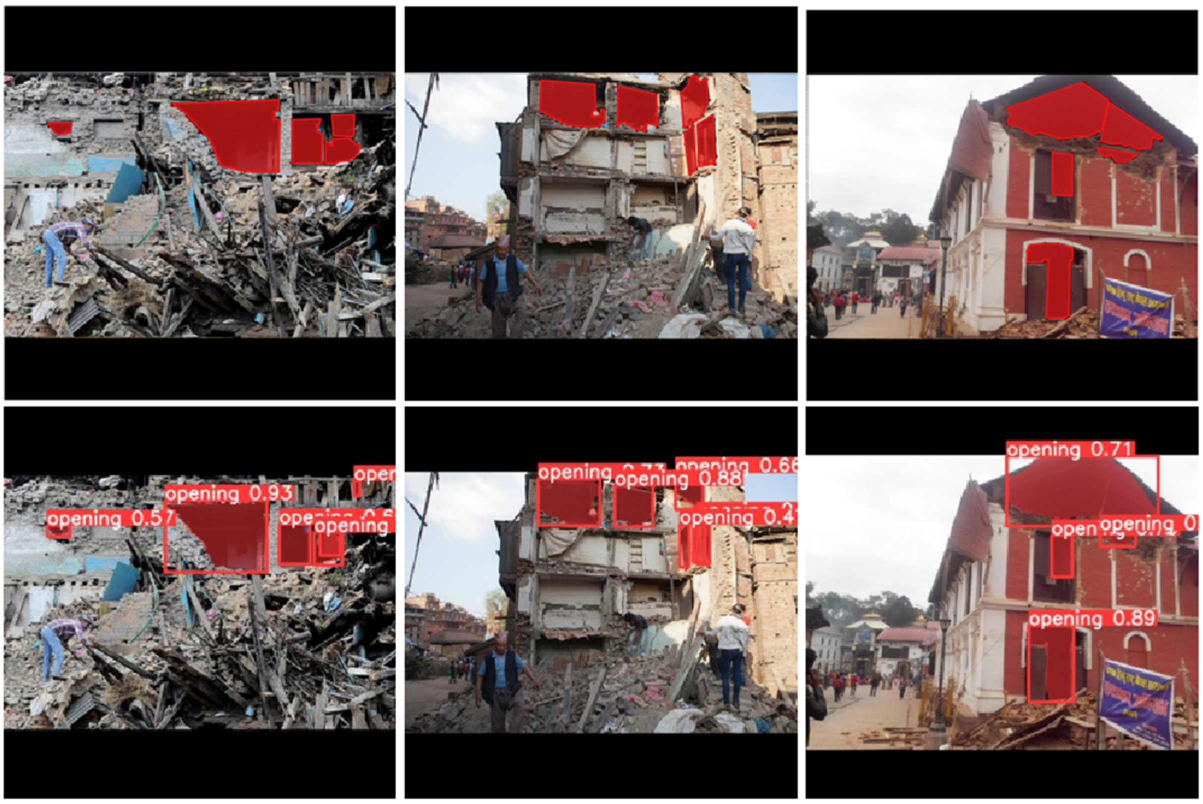

Figure 6. On-the-ground reality (**above**) and YOLOv5s-MTL predictions (**below**) on different types of damaged building openings.

Collapsed walls or roofs were detected and segmented quite well by this model. As shown in Figure 6, different sizes and forms of openings due to collapsed walls and roofs were able to be detected and segmented by this YOLOv5s-MTL model. Open doors and windows were also detected and segmented. The predicted mask on the test images showed that the model correctly delineated the opening space area. Qualitatively, the multitask learning YOLOv5 model also showed better detection results compared to the single-task detector based on visual interpretation.

5.3. Single and MTL Model Performance Comparison

In the previous sections, single and multitask learning-based detection networks were trained on the damaged building-opening dataset. This section will compare both learning approaches and evaluate each task's benefits and drawbacks for the damaged building-opening detection scenario. The model performance comparison is discussed as follows.

The detection performances of single-task (YOLOv5s) and multitask (YOLOv5s-MTL) are listed in Table 3. YOLOv5s-MTL showed improvement in precision, recall, and mean-average precision. While the single task YOLOv5s outperformed in terms of inference speed, the YOLOv5s-MTL still satisfied the FPS requirement for real-time application.

Table 3. YOLOv5 vs YOLOv5-MTL detection performance comparison.

Model	Precision	Recall	mAP50	Speed (FPS)
YOLOv5s	0.609	0.477	0.481	107
YOLOv5s-MTL	0.747 (+0.138)	0.577 (+0.1)	0.648 (+0.167)	73 (−34)

The results showed that adding the segmentation head on the network can improve detection accuracy but decrease inference speed. The idea of combining detection and segmentation loss was proven to be able to improve the model's mAP by 0.167. Multitask learning tends to have a slower inference speed than single-task models due to a larger number of parameters. However, the inference speed of YOLOv5s-MTL can still be categorized as real-time (>30 FPS).

Figure 7 compares the on-the-ground reality image, detection output from the YOLOv5s model, and output from YOLOv5s-MTL, respectively. This figure shows that multitask learning improved the model's accuracy. The openings not fully detected by the single-task model were comprehensively detected and segmented by the multitask learning based network.

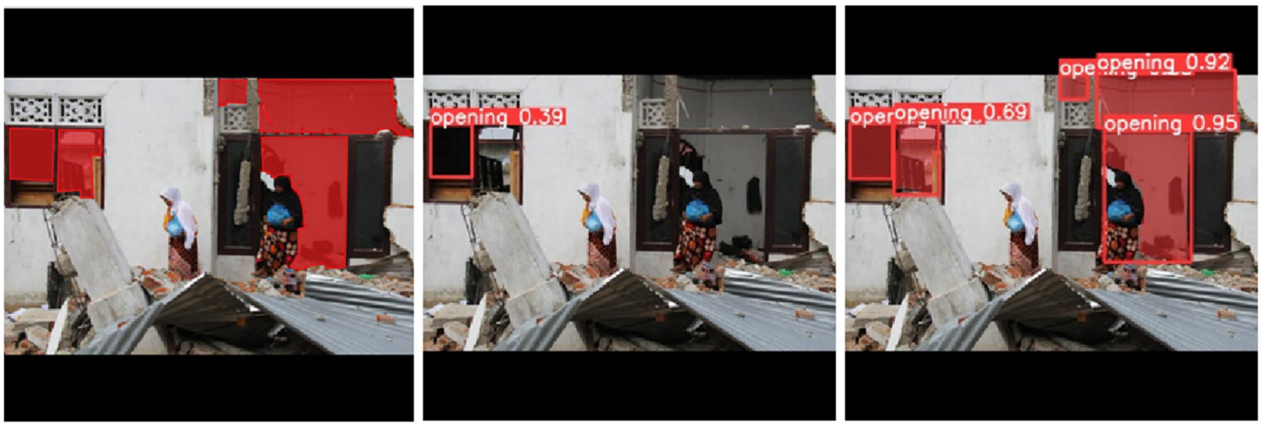

Figure 7. On-the-ground reality image, single-task, and multitask output (left to right).

In terms of precision, YOLOv5s-MTL was better than YOLOv5s. From Figure 8, it can be observed that multitask learning had fewer false positive detections than single-task networks. The detection errors are highlighted in orange circles. Overall, the multitask model had fewer errors than the single task. Although the multitask model prediction results also showed some detection errors, these results indicate that the better contextualization given by the image segmentation helped the detection task to reduce uncertainties in identifying openings.

Based on visual interpretation of the detection results, the YOLOv5s-MTL showed improvement by decreasing the false positives and false negatives in the single-task detectors. This MTL network alleviated the background classification mechanism problems in the single-task model (Figure 8). The segmentation mask provided fitted boundaries that separated objects and backgrounds. To decrease false positive detections, the image segmentation task helped the model to contextualize better.

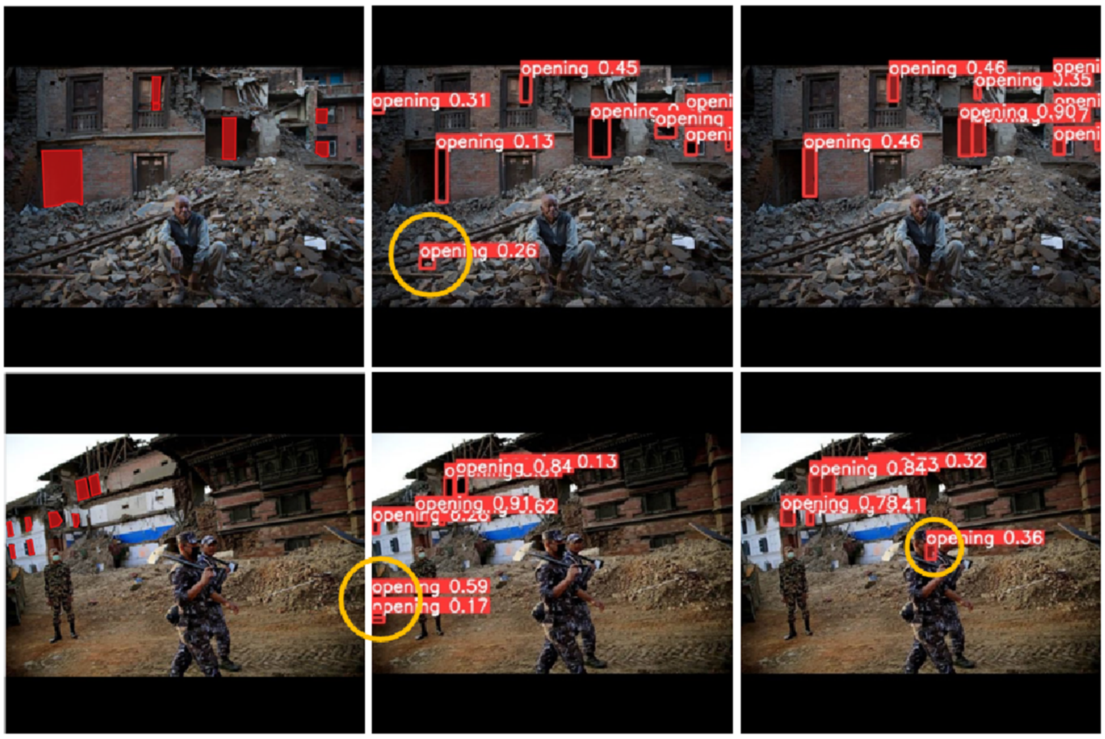

Figure 8. On-the-ground reality image, single-task, and multitask output (left to right), showing results of YOLOv5-MTL reducing false positives.

5.4. Performance Comparison with SOTA Detection Model

As part of the experiments, this study trained the recent versions of the YOLO model and compared its performance with the proposed YOLOv5 multitask learning model. These experiments were conducted on a damaged building-opening dataset and models were trained for 100 epochs with early stopping.

As shown in Table 4, YOLOv5s-MTL outperformed the other models in terms of precision, recall, and mAP50. It had the highest values for all three metrics, indicating better overall detection performance. Newer versions of the YOLO model, specifically YOLOv7 and YOLOv8, do not guarantee a consistent increase in detection performance compared to their predecessor, YOLOv5. YOLOv7, despite being a newer version, exhibited slightly lower precision and mAP50 values than YOLOv5s-MTL, implying that the advancements in the model architecture may not necessarily translate into across-the-board improvements. Similarly, YOLOv8s, another newer iteration, showcased a decent precision but fell short in terms of recall and mAP50 compared to YOLOv5s (both the single and multitask learning models). This discrepancy suggests that the pursuit of higher model versions does not automatically ensure enhanced detection capabilities, and the performance gains may vary or even diminish in certain aspects.

Table 4. Detection performance comparison with current state-of-the-art real-time object detection models.

Model	Precision	Recall	mAP50
YOLOv5s	0.609	0.477	0.481
YOLOv5s-MTL	**0.747**	**0.577**	**0.648**
YOLOv7	0.564	0.510	0.510
YOLOv8s	0.595	0.444	0.469
YOLOv8s-MTL	0.553	0.467	0.478

6. Discussion

6.1. Damaged Building-Opening Dataset Development

This study developed a dataset of images and corresponding annotations for a damaged building-opening detection model encompassing open doors/windows, collapsed walls, and damaged roofs. To simplify the detection model, all opening types were categorized into one class, (i.e., openings), as in a real-life application scenario there is no need to classify the opening types. This simplifies problems in the case of a single-class detection scenario.

The developed dataset was sufficient for training a damaged building-opening detection model. However, some aspects were not covered: (i) the ground sampling distance (GSD) of the images in this dataset was not uniform. A known image GSD can guarantee to link an area in an image to a certain size in the object space, supporting the model's performance in detecting various sizes of openings; the images used in this study were obtained from open data sources and captured from different cameras with different settings. (ii) There is no publicly available open data source for damaged building-facade scenes captured using UAV. Therefore, the images used in this dataset were filtered to select the most relevant images representing a point of view similar to actual scenes of damaged buildings captured by a UAV. In addition, the different openings were not balanced, privileging conventional openings (e.g., windows and doors) over damage-induced openings (e.g., holes in the façade).

6.2. Damaged Building Opening Detection

The study found that a single-task YOLOv5 object detection network trained on a damaged building-opening dataset can perform the detection quite well. From the experiments training on three different versions of YOLOv5, we found that the models' performances can vary although the larger version of YOLOv5 did not deliver much higher accuracy. We can see from the performance of the YOLOv5l (large) that it was similar to the medium version (YOLOv5m): intuitively, a larger model would be expected to obtain better accuracy, but the reduced training size probably had negative impacts on the more complex networks. The mAP50 of YOLOv5m was slightly better than YOLOv5l, with a 0.011 difference: this could be because of the effect on the number of convolutional kernels in YOLOv5l; too many convolution kernels on the network can sometimes result in overfitting. In the small version of YOLOv5, the bounding box score prediction already showed subtle signs of overfitting; this became worse in deeper networks, as in the large version of YOLOv5. For this reason, we needed to find the best-fitting model architecture that had a better performance metric, but still satisfied the FPS requirement for real-time applications. The findings showed that increasing the model complexity does not guarantee improvement in the model performance. In this case, we found that the medium version of YOLOv5 was the optimal model configuration for a single-task damaged building-opening detector.

Despite the good performance in detecting the opening spaces of a damaged building, several errors could still be found in the detections. As illustrated in Figure 5, false negatives and false positives still appeared in the detection results. False negatives were probably due to the background classification mechanism. False positives are not desirable for damaged building opening detection: if the model detects a non-opening as an opening, there will be a failure to enter a damaged building. Thus, it becomes crucial to minimize false positive detections.

The multitask learning detection model partially overcame this problem by outperforming the single-task-based model with the same training. From these results, it can be noticed that the more complex the model is in YOLOv5-MTL, the more the model's accuracy increases, showing fewer overfitting problems: YOLOv5l-MTL showed higher mAP50 than two other smaller model versions.

The multitask learning model had, however, a slower inference time than the single-task detectors. This happened because working with two tasks will increase the model's complexity, thus requiring more processing time. YOLOv5s-MTL performed the fastest inference time, with 73 FPS tested on an NVIDIA Tesla T4 GPU, while the YOLOv5l-MTL gave the best accuracy, but the inference time of this model was slow, with only 27 FPS. This means that the large version of YOLOv5-MTL cannot perform real-time inference (<30 FPS) in current settings. Conversely, YOLOv5m-MTL barely met the real-time inference requirements with 30 FPS. The drop in inference speed, when switching from a small version of single-task YOLOv5 to a multitask one, was lower compared to increasing the size of the base network to the medium version (i.e., from YOLOv5s-MTL to YOLOv5m), while the corresponding increase in accuracy was significantly higher (i.e., 0.167 for YOLOv5s-MTL and only 0.02 for YOLOv5m). In that regard, YOLOv5s-MTL seems to be the best balance between performance and inference speed. In general, the multitask learning model performed better than the single-task detector while still having real-time inference.

The YOLOv5 algorithm has shown good performances on edge devices, as stated by the wider scientific literature. In this paper, we assessed the drop in the FPS from the conventional algorithm to the YOLOv5-MTL algorithm. Our tests showed that the FPS reduction is relatively limited (despite the increased number of parameters) and it is still usable on state-of-the-art edge devices for real-time processing.

Large language models and large visual models, as well as visual question and answering (VQA) algorithms [45,46], will drastically change the use of UAVs for understanding scenes in real-time: the "conversion" of object search and detection on images into textual messages will allow for a more direct interaction between UAVs and their operators in the field. Future research will be undertaken to explore these innovative solutions.

7. Conclusions and Future Developments

The main goal of this study was to develop a deep learning detection model for real-time damaged building-opening detection. Open doors or windows, damaged walls, and collapsed roofs are defined as damaged building openings. To train the detection model, a damaged building-opening dataset was developed. The process of dataset development consisted of image selection, annotation, pre-processing, and augmentations.

The conventional YOLOv5, and a multitask version combining object detection and segmentation tasks, were trained and evaluated on the damaged building-opening dataset. In particular, three different versions of the algorithms, i.e., small, medium, and large, were compared as detector models for damaged building openings. This study found that false positives and false negatives can be reduced by implementing the multitask learning method. The proposed multitask learning model sacrificed the processing time slightly but achieved better accuracy. Compared to the single-task detector, the YOLOv5-MTL improved the model's mAP50 by about 0.167. The multitask approach required a higher number of parameters and longer processing time compared to the conventional YOLO implementation. The FPS reduction is, however, limited and it still allows its deployment on edge devices.

Overall, this study contributes to developing a novel dataset, and sheds some light on deep learning-based detectors for damaged building-opening detection. It also demonstrates that a multitask learning method can improve detection accuracy. Lastly, it broadens our knowledge in developing a real-time deep learning-based detection-model algorithm. With more rapid development of UAV applications in terms of systems and data acquisitions, there is potential to improve practices in disaster response applications. Several possible improvements can be considered for future research development, such as training dataset improvement, and assessment of the introduction of a depth estimation head in the multitask learning approach for a damaged building-opening detection model.

Author Contributions: A.S.: Data Curation, Investigation, Methodology, Formal analysis, Visualization, and Writing—Original Draft. N.Z.: Supervision, Software, Resources, Methodology, and Writing—Review and Editing. J.R.B.: Supervision, Resources, Methodology, Validation, and Writing—Review and Editing. F.N.: Supervision, Conceptualization, Methodology, Project administration, and Writing—Review and Editing. Each author contributed significantly to the development and execution of the research project. A.S. played a key role in data curation, performing the experiments, conducting formal analyses, and drafting the initial manuscript. N.Z. contributed to the software used to perform the experiments, the design and development of the methods, the resources which the methods are based on, and provided input during the review and editing stages. J.R.B. contributed resources, the design and development of the methods, validated results, and participated in the review and editing process. N.Z., J.R.B. and F.N. provided overall supervision for the project. F.N. conceptualized the study, managed project administration, and contributed to the final review and editing of the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Publicly available datasets were analyzed in this study. This dataset can be found here: https://github.com/UAV-Centre-ITC/yolo-damaged-building-opening-detection (accessed on 25 January 2024).

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

References

1. Mittal, M.; Mohan, R.; Burgard, W.; Valada, A. Vision-Based Autonomous UAV Navigation and Landing for Urban Search and Rescue. In *Robotics Research*; Asfour, T., Yoshida, E., Park, J., Christensen, H., Khatib, O., Eds.; Springer Proceedings in Advanced Robotics; Springer International Publishing: Cham, Switzerland, 2022; Volume 20, pp. 575–592, ISBN 978-3-030-95458-1.
2. Delmerico, J.; Mintchev, S.; Giusti, A.; Gromov, B.; Melo, K.; Horvat, T.; Cadena, C.; Hutter, M.; Ijspeert, A.; Floreano, D.; et al. The Current State and Future Outlook of Rescue Robotics. *J. Field Robot.* **2019**, *36*, 1171–1191. [CrossRef]
3. Gomez, C.; Purdie, H. UAV- Based Photogrammetry and Geocomputing for Hazards and Disaster Risk Monitoring–A Review. *Geoenviron. Disasters* **2016**, *3*, 23. [CrossRef]
4. Goian, A.; Ashour, R.; Ahmad, U.; Taha, T.; Almoosa, N.; Seneviratne, L. Victim Localization in USAR Scenario Exploiting Multi-Layer Mapping Structure. *Remote Sens.* **2019**, *11*, 2704. [CrossRef]
5. Zhang, N.; Nex, F.; Vosselman, G.; Kerle, N. Training a Disaster Victim Detection Network for UAV Search and Rescue Using Harmonious Composite Images. *Remote Sens.* **2022**, *14*, 2977. [CrossRef]
6. Nex, F.; Duarte, D.; Steenbeek, A.; Kerle, N. Towards Real-Time Building Damage Mapping with Low-Cost UAV Solutions. *Remote Sens.* **2019**, *11*, 287. [CrossRef]
7. Zhang, N.; Nex, F.; Kerle, N.; Vosselman, G. LISU: Low-Light Indoor Scene Understanding with Joint Learning of Reflectance Restoration. *ISPRS J. Photogramm. Remote Sens.* **2022**, *183*, 470–481. [CrossRef]
8. Zhao, W.; Zhang, W.; Pan, X.; Zhuang, P.; Xie, X.; Li, L.; Yang, H. LESSL: Can LEGO Sampling and Collaborative Optimization Contribute to Self-Supervised Learning? *Inf. Sci.* **2022**, *615*, 475–490. [CrossRef]
9. Alsamhi, S.H.; Shvetsov, A.V.; Kumar, S.; Shvetsova, S.V.; Alhartomi, M.A.; Hawbani, A.; Rajput, N.S.; Srivastava, S.; Saif, A.; Nyangaresi, V.O. UAV Computing-Assisted Search and Rescue Mission Framework for Disaster and Harsh Environment Mitigation. *Drones* **2022**, *6*, 154. [CrossRef]
10. Lyu, M.; Zhao, Y.; Huang, C.; Huang, H. Unmanned Aerial Vehicles for Search and Rescue: A Survey. *Remote Sens.* **2023**, *15*, 3266. [CrossRef]
11. Półka, M.; Ptak, S.; Kuziora, Ł. The Use of UAV's for Search and Rescue Operations. *Procedia Eng.* **2017**, *192*, 748–752. [CrossRef]
12. Barbosa, J.; Hernandez, C.; Paredes, D.; Jativa, E.R. Design and Implementation of an Autonomous Vehicle with LIDAR-Based Navigation. In Proceedings of the 2020 International Conference on Mechatronics, Electronics and Automotive Engineering (ICMEAE), Cuernavaca, Mexico, 16–21 November 2020; IEEE: Cuernavaca, Mexico, 2020; pp. 98–103.
13. Aldao, E.; González-de Santos, L.; González-Jorge, H. LiDAR Based Detect and Avoid System for UAV Navigation in UAM Corridors. *Drones* **2022**, *6*, 185. [CrossRef]
14. Zhang, Z.; Zhu, L. A Review on Unmanned Aerial Vehicle Remote Sensing: Platforms, Sensors, Data Processing Methods, and Applications. *Drones* **2023**, *7*, 398. [CrossRef]
15. Elamin, A.; Abdelaziz, N.; El-Rabbany, A. A GNSS/INS/LiDAR Integration Scheme for UAV-Based Navigation in GNSS-Challenging Environments. *Sensors* **2022**, *22*, 9908. [CrossRef] [PubMed]
16. Petrlik, M.; Krajnik, T.; Saska, M. LIDAR-Based Stabilization, Navigation and Localization for UAVs Operating in Dark Indoor Environments. In Proceedings of the 2021 International Conference on Unmanned Aircraft Systems (ICUAS), Athens, Greece, 15–18 June 2021; IEEE: Athens, Greece, 2021; pp. 243–251.

17. Pritzl, V.; Stepan, P.; Saska, M. Autonomous Flying into Buildings in a Firefighting Scenario. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 239–245.
18. Arafat, M.Y.; Alam, M.M.; Moh, S. Vision-Based Navigation Techniques for Unmanned Aerial Vehicles: Review and Challenges. *Drones* **2023**, *7*, 89. [CrossRef]
19. Kumar, R.H.; Vanjare, A.M.; Omkar, S.N. Autonomous Drone Navigation Using Monocular Camera and Light Weight Embedded System. In Proceedings of the 2023 International Conference for Advancement in Technology (ICONAT), Goa, India, 24–26 January 2023; IEEE: Goa, India, 2023; pp. 1–6.
20. Sandino, J.; Vanegas, F.; Maire, F.; Caccetta, P.; Sanderson, C.; Gonzalez, F. UAV Framework for Autonomous Onboard Navigation and People/Object Detection in Cluttered Indoor Environments. *Remote Sens.* **2020**, *12*, 3386. [CrossRef]
21. Matsuzaka, Y.; Yashiro, R. AI-Based Computer Vision Techniques and Expert Systems. *AI* **2023**, *4*, 289–302. [CrossRef]
22. Zou, Z.; Chen, K.; Shi, Z.; Guo, Y.; Ye, J. Object Detection in 20 Years: A Survey. *Proc. IEEE* **2023**, *111*, 257–276. [CrossRef]
23. Zhao, Z.-Q.; Zheng, P.; Xu, S.; Wu, X. Object Detection with Deep Learning: A Review. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *30*, 3212–3232. [CrossRef] [PubMed]
24. Ramôa, J.G.; Lopes, V.; Alexandre, L.A.; Mogo, S. Real-Time 2D-3D Door Detection and Classification on Jetson Nano. *SN Appl. Sci.* **2021**, *3*, 590. [CrossRef] [PubMed]
25. Yu, C.; Wang, J.; Peng, C.; Gao, C.; Yu, G.; Sang, N. BiSeNet: Bilateral Segmentation Network for Real-Time Semantic Segmentation. In Proceedings of the European Conference on Computer Vision (ECCV 2018), Munich, Germany, 8–14 September 2018. [CrossRef]
26. Balamuralidhar, N.; Tilon, S.; Nex, F. MultEYE: Monitoring System for Real-Time Vehicle Detection, Tracking and Speed Estimation from UAV Imagery on Edge-Computing Platforms. *Remote Sens.* **2021**, *13*, 573. [CrossRef]
27. Abdulateef, S.; Salman, M. A Comprehensive Review of Image Segmentation Techniques. *Iraqi J. Electr. Electron. Eng.* **2021**, *17*, 166–175. [CrossRef]
28. Minaee, S.; Boykov, Y.Y.; Porikli, F.; Plaza, A.J.; Kehtarnavaz, N.; Terzopoulos, D. Image Segmentation Using Deep Learning: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 3523–3542. [CrossRef] [PubMed]
29. Jocher, G.; Chaurasia, A.; Stoken, A.; Borovec, J.; NanoCode012; Kwon, Y.; Michael, K.; Tao, X.; Fang, J.; Imyhxy; et al. Ultralytics/Yolov5: V7.0-YOLOv5 SOTA Realtime Instance Segmentation. Available online: https://zenodo.org/records/7347926 (accessed on 25 January 2024).
30. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 779–788.
31. Yan, B.; Fan, P.; Lei, X.; Liu, Z.; Yang, F. A Real-Time Apple Targets Detection Method for Picking Robot Based on Improved YOLOv5. *Remote Sens.* **2021**, *13*, 1619. [CrossRef]
32. Wang, C.-Y.; Liao, H.-Y.M.; Yeh, I.-H.; Wu, Y.-H.; Chen, P.-Y.; Hsieh, J.-W. CSPNet: A New Backbone That Can Enhance Learning Capability of CNN 2019. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Seattle, WA, USA, 14–19 June 2020.
33. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection 2020. *arXiv* **2020**, arXiv:2004.10934.
34. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. In *Computer Vision–ECCV 2014*; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2014; Volume 8691, pp. 346–361, ISBN 978-3-319-10577-2.
35. Zhang, W.; Zhao, W.; Li, J.; Zhuang, P.; Sun, H.; Xu, Y.; Li, C. CVANet: Cascaded Visual Attention Network for Single Image Super-Resolution. *Neural Netw.* **2024**, *170*, 622–634. [CrossRef] [PubMed]
36. Crawshaw, M. Multi-Task Learning with Deep Neural Networks: A Survey. *arXiv* **2020**, arXiv:2009.09796. [CrossRef]
37. Vandenhende, S.; Georgoulis, S.; Van Gansbeke, W.; Proesmans, M.; Dai, D.; Van Gool, L. Multi-Task Learning for Dense Prediction Tasks: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 3614–3633. [CrossRef]
38. He, K.; Gkioxari, G.; Dollar, P.; Girshick, R. Mask R-CNN. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2980–2988.
39. Bolya, D.; Zhou, C.; Xiao, F.; Lee, Y.J. YOLACT: Real-Time Instance Segmentation 2019. In Proceedings of the 2019 IEEE/CVF In-ternational Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019.
40. Elfwing, S.; Uchibe, E.; Doya, K. Sigmoid-Weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning. *Neural Netw.* **2018**, *107*, 3–11. [CrossRef]
41. Kumar, A.; Rawat, Y.S. End-to-End Semi-Supervised Learning for Video Action Detection. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 14680–14690.
42. Chachra, G.; Kong, Q.; Huang, J.; Korlakunta, S.; Grannen, J.; Robson, A.; Allen, R.M. Detecting Damaged Buildings Using Real-Time Crowdsourced Images and Transfer Learning. *Sci. Rep.* **2022**, *12*, 8968. [CrossRef]
43. Henderson, P.; Ferrari, V. End-to-End Training of Object Class Detectors for Mean Average Precision. In *Computer Vision–ACCV 2016*; Lai, S.-H., Lepetit, V., Nishino, K., Sato, Y., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2017; Volume 10115, pp. 198–213, ISBN 978-3-319-54192-1.
44. Singha, T.; Pham, D.-S.; Krishna, A. A Real-Time Semantic Segmentation Model Using Iteratively Shared Features in Multiple Sub-Encoders. *Pattern Recognit.* **2023**, *140*, 109557. [CrossRef]

45. De Curtò, J.; De Zarzà, I.; Calafate, C.T. Semantic Scene Understanding with Large Language Models on Unmanned Aerial Vehicles. *Drones* **2023**, *7*, 114. [CrossRef]
46. Chen, C.; Zheng, Z.; Xu, T.; Guo, S.; Feng, S.; Yao, W.; Lan, Y. YOLO-Based UAV Technology: A Review of the Research and Its Applications. *Drones* **2023**, *7*, 190. [CrossRef]