applied sciences

MDPI

Article

Dist-YOLO: Fast Object Detection with Distance Estimation

Marek Vajgl [1,*], Petr Hurtik [1] and Tomáš Nejezchleba [2]

[1] Centre of Excellence IT4Innovations, Institute for Research and Applications of Fuzzy Modeling, 30. Dubna 22, University of Ostrava, 702 00 Ostrava, Czech Republic; petr.hurtik@osu.cz

[2] Varroc Lighting Systems, Suvorovova 195, 742 42 Šenov u Nového Jičína, Czech Republic; tnejezch@varroclighting.com

* Correspondence: marek.vajgl@osu.cz; Tel.: +420-553-46-1414

Abstract: We present a scheme of how YOLO can be improved in order to predict the absolute distance of objects using only information from a monocular camera. It is fully integrated into the original architecture by extending the prediction vectors, sharing the backbone's weights with the bounding box regressor, and updating the original loss function by a part responsible for distance estimation. We designed two ways of handling the distance, class-agnostic and class-aware, proving class-agnostic creates smaller prediction vectors than class-aware and achieves better results. We demonstrate that the subtasks of object detection and distance measurement are in synergy, resulting in the increase of the precision of the original bounding box functionality. We show that using the KITTI dataset, the proposed scheme yields a mean relative error of 11% considering all eight classes and the distance range within [0, 150] m, which makes the solution highly competitive with existing approaches. Finally, we show that the inference speed is identical to the unmodified YOLO, 45 frames per second.

Keywords: monocular camera; YOLO; distance estimation; object detection

check for updates

Citation: Vajgl, M.; Hurtik, P.; Nejezchleba, T. Dist-YOLO: Fast Object Detection with Distance Estimation. *Appl. Sci.* **2022**, *12*, 1354. https://doi.org/10.3390/app12031354

Academic Editors: Deokwoo Lee and YoHan Park

Received: 5 January 2022
Accepted: 24 January 2022
Published: 27 January 2022

1. Introduction

Distance estimation is an essential part of 3D scene recognition and orientation, allowing various autonomous devices to move in the natural environment. These devices can be equipped with passive capturing devices, such as RGB or IR cameras, and active capturing devices, including lidar or sonar. Active devices are more expensive but provide distance estimation directly as a cloud of points [1], while passive devices require complex computer vision algorithms on top to estimate the distance [2].

In this study, we designed a computer vision algorithm for passive capturing devices. The algorithm detects objects in a scene together with the information about the absolute distance from a camera. Such an algorithm can be applied in automatic dimming/brightening of cars' headlamps to prevent the dazzling of other cars or to highlight various obstacles on the road. For this task, a scene in front of the car is captured by an RGB camera, the objects of interest (cars/people/animals) are localized, and a control unit of a headlamp adjusts particular light-emitting segments in order not to dazzle the object, or highlight it. The decision algorithm needs the location and the distance to an object to make a correct decision. Furthermore, the processing of such data has to be able to run in realtime on an embedded device. The usage of an RGB camera instead of an active capturing device makes the solution significantly cheaper and extends the visibility range because a camera can see further than a sonar or lidar [3].

Our idea is motivated by human perception, where we can recognize objects in a scene and estimate their distance just because we can create reasoning such as 'This is a bicycle, and it is small, so it is approximately 40 m far'. For such reasoning, a pair of eyes helps, but the ability is partially preserved even if we will use only one eye. In image processing, we can estimate a distance for every pixel in a scene or per object. The 'per object' case estimation corresponds to the mentioned human perception. This study discusses how an object detection and recognition algorithm can be naturally extended for distance estimation reflecting native human perception, that is, for a system where visual information from a monocular camera is available only. For this case, one of the fastest one-stage object detection algorithms, YOLOv3 [4], has been selected, but the proposed principle can be used also in YOLOv4 [5] and YOLOv5 [6], as they have the same inner structure. The goal of the work is to propose an extension that will not lead to an increased computational cost compared to the original algorithm, while the distance estimation quality will be comparable to other state-of-the-art systems. This study aims to create a method for estimating *the absolute distance* (instead of relative distance/depth information) to set light beams correctly.

1.1. Related Work

The existing methods can be split into two groups, predicting relative and absolute distance. The methods for estimating relative distance can express a particular object's (given by a bounding box) distance on a relative scale [7] or realize dense prediction and produce a depth map for the entire image. The depth map can be entirely independent between distinct images [8] or preserve the consistency [9]. We can view the depth map as a type of disparity map, where it can be produced using a single image only [10] or via multiple images [11].

The methods predicting absolute distance (in meters, inches, etc.) can be further separated into those that use data from active devices, passive devices, or their combination.

Active devices are mostly represented by lidar, which has a better visibility range than sonar. VoxelNet [12] is trained in the end-to-end scheme over the lidar raw point cloud and can detect objects of car, cyclist, and pedestrian classes together with their precise 3D bounding boxes and position in a 3D scene. Study [13] proposed coupling two VoxelNets, one for region proposal and the second one for patch refinement, which led to a precision improvement on the KITTI 3D object detection benchmark [14,15].

Regarding the combination of both types of devices, we can find models trained using lidar and visual data but perform inference using visual information only. Saxena et al. [16] separated an RGB image into small patches and by applying a probabilistic model produced a depth map close to the output of a 3D scanner. Generally, such approaches search for optimal mapping from the visual domain into a depth map, which is ideal for deep neural networks based on autoencoders [17]. Analysis [18] demonstrated that object-specific distance estimation using a monocular camera was not flawless, so they segmented Velodyne point clouds and fused them with features from a model to precisely predict the distance.

Regarding passive devices, Zhang et al. [3] used three cameras with a small field of view and using a disparity calculation produced a dense depth map with reasonable results even for an object with distance higher than 200 m. The crucial point is to deal with pseudo-rectification and ambiguity removal.

Hu et al. [19] used a monocular camera and performed object detection and tracking, where the distance of an object was identified as valuable information for the tracker. The idea was based on Faster R-CNN enhanced by distance and angle estimation. Study [20] aimed at 3D box detection and an angle estimation network, which provided attitude angle information, and using the camera projection principle over this information, the distance was precisely determined. Natanael et al. [21] utilized YOLOv3 to detect bounding boxes together with coordinates, and from them, computed distances analytically. DisNet [2,22] combined YOLOv3 that produced bounding box (BB) coordinates and a fully connected neural network that produced distances. The fully connected network was trained separately on predictions taken from YOLO. Chen et al. [23] used YOLOv3 and coupled it with Monodepth [24], i.e., a model trained on visual information from two cameras to produce a disparity map on a single camera during inference. The distance was estimated from Monodepth's output and injected into YOLO's predicted boxes. Strbac et al. [25] used two cameras and two YOLOv3 detectors, where the distance was measured with a stereoscopic principle over the detected boxes. The disadvantages were the doubled computation time due to the two detectors and the impossibility of determining the distance when only one detector found an object. Mauri et al. [26] integrated into YOLOv3 a 3D regression module that took the detected boxes together with feature maps and utilized additional convolution layers to produce the orientation, dimension, and distance of each of the boxes. Because the 3D regression module works on the produced bounding box coordinates, it turned the original one-stage YOLO principle into a two-stage one, which is different from our proposal.

To date, no solution fully integrates distance estimation into the YOLO architecture to measure distance using a monocular camera. Such a solution is the aim of this study.

1.2. The Motivation and Our Idea in Brief

Our long-term goal is to create a system capturing a scene in front of a car and providing information about the detected objects together with their *absolute* distances. The system has to be inexpensive; therefore, it does not rely on lidar, and the object detector runs on a low-powered embedded device, Jetson Nano. The proposed system is a part of a project controlling LED segments in a car's headlamps to highlight detected objects. To set up the light beam precisely, absolute distance information is necessary, so it is not feasible to use methods providing relative distance, e.g., dense depth map [7,9].

We use a monocular RGB camera. The motivation to omit two cameras and compute a disparity map for distance estimation is that according to [27] "A single-pixel error in disparity implies only a 0.1m error in depth at a depth of 5 m, but a 5.8 m error at a depth of 50 m" on a KITTI dataset [14], and the current object detectors have a more significant pixel error. Moreover, we are restricted to a monocular camera because the usage of two cameras increases the price of the solution.

Our intuition is that YOLO performs regression on BB coordinates, so it can be trained to solve the regression problem of absolute distance estimation as well. The assumption is that the network can use the same inner features for BB coordinates and distance estimation, leading to synergy. It motivates us to integrate the part responsible for distance estimation into the YOLO architecture and train the model in an end-to-end manner. The assumption will be confirmed when a model trained for distance estimation leads to higher BB precision than the baseline YOLO.

Our contributions to the problem are as follows:

1. We define a novel architecture, Dist-YOLO, where prediction vectors produced by heads are extended by information about distance and coupled with a proper distance loss function.
2. We show that Dist-YOLO detects bounding boxes more accurately than the original YOLO while having the same backbone's capacity.
3. We demonstrate that a monocular camera with Dist-YOLO can precisely estimate the distance of an object.

2. Dist-YOLO

YOLO is used to detect an object's bounding box and classify the object into one of the predefined classes. By default, it cannot estimate the distance of the object. Our primary goal is to add this ability to YOLO to create Dist-YOLO, while preserving the original properties. To achieve this capability, three basic steps must be followed:

(a) Enrich the labels in the training dataset with information about distances.
(b) Extend the prediction in each cell to produce the distance of an object.
(c) Update the YOLOv3 loss function used for training to take into account the distance of an object.

2.1. Preliminaries-YOLOv3

YOLO—*You Only Look Once*—is a one-stage, multi-scale, anchor-based object detector utilizing a fully convolutional architecture, DarkNet-53. Initially introduced by Joseph Redmon [28] and evolved by them into the second and third version [4,29], where the first version utilized single scale detection and lightweight architecture. Currently, the fourth [5] and fifth versions have been introduced by independent groups. The performance boost of the fourth and fifth versions is achieved mainly by new data augmentation and minor architecture changes; the main principle remains the same [5]. The newest version, YOLOX [30] is based on YOLOv3 and modifies it into an anchor-free algorithm which allows use of decoupled heads, leading to a slightly higher accuracy of detection. Furthermore, YOLO has been extended in several ways to improve its precision and capabilities. Namely, object segmentation based on object polygon detection, Poly-YOLO, was introduced in [31], where the original YOLO prediction vector was extended to predict a set of polygon points together with their confidence for every detected object. Furthermore, an approach motivated by a Mask R-CNN $\times$ Faster R-CNN relation, YOLACT, was introduced in [32] to provide the ability for mask instance segmentation with comparable results to standard two-staged segmentation approaches, but improving their computational demands. Later, this approach was extended into YOLACT++ [33] with additional performance and precision boost.

In contrast to two-staged detectors [34], YOLO does not use a region proposal network but integrates the whole process into a single architecture. It is split into encoder and decoder parts. The encoder serves as a feature extractor. It is represented by the Darknet-53 backbone, but an arbitrary backbone can be used. The features are decoded on three succeeding scales into three output grids. Each grid consists of cells, where each cell is responsible for detecting objects whose center lies inside the cell. By detecting, we mean the regression on the bounding box coordinates together with class and confidence. In YOLOv3, each cell can detect three objects, where so-called anchors are used for suppressing the problem of detecting three identical objects. By anchor, we mean a prototypical bounding box extracted from training data during preprocessing using k-means algorithm. During training and inference, a cell is assigned three anchors and detects an object in the output position where the intersect over union (IoU) over the box and the anchor is maximized. Compared to two-stage detectors, YOLO is much faster but yields lower precision of detection.

The success of the YOLO algorithm family can be illustrated on a wide range of tasks where it was applied, such as face mask wearing detection [35], identification of plant diseases [36], pedestrian detection [37], forest fire detection [38], or safety helmet detection [39]. Currently, YOLO based approaches are used for popular social distancing estimation [40,41], also combined with mask detection [42].

2.2. Updating the Predictions Vector

A prediction vector p of cell and anchor at a certain scale is given as $\mathbf{p} = (\mathbf{b}, \mathbf{c}, o)$, where $\mathbf{b} = (x, y, w, h)$ are coordinates produced by a bounding box regressor, $\mathbf{c} = (c_1, c_2, \dots, c_n)$ that addresses the confidence of being a certain class up to n classes, and o expresses objectness, i.e., confidence that $\mathbf{p}$ captures a real object. In total, each prediction vector consists of $5 + n$ values, and we have $3 \times 3 \times r$ prediction vectors, where r is the number of all cells in the three output layers. Typically, r = 14,157.

Regarding distance, we extend the prediction vector into form $\mathbf{p} = (\mathbf{b}, \mathbf{c}, o, \mathbf{d})$, where we have two options of how to define $\mathbf{d}$. Firstly, $\mathbf{d} = (d)$ if the distance is class-agnostic and secondly, $\mathbf{d} = (d_1, d_2, \dots, d_n)$ if the distance is class-aware. The prediction vector is produced by 1×1 convolutional filters, where each of the filters produces a single value representing a certain variable, e.g., x coordinate of the first object, class of the second object, etc. Therefore, when we add additional information (distance), it is necessary to increase the total number of these filters. Each filter includes one trainable parameter, because architectures implementing batch normalization [43], including YOLO, do not use bias. If we consider a typical r and, e.g., $n = 7$, class-agnostic distance increases the number of trainable parameters in the output layers by 8%, and class-aware distance increases them by 58%. The illustration scheme of the Dist-YOLO architecture is shown in Figure 1.

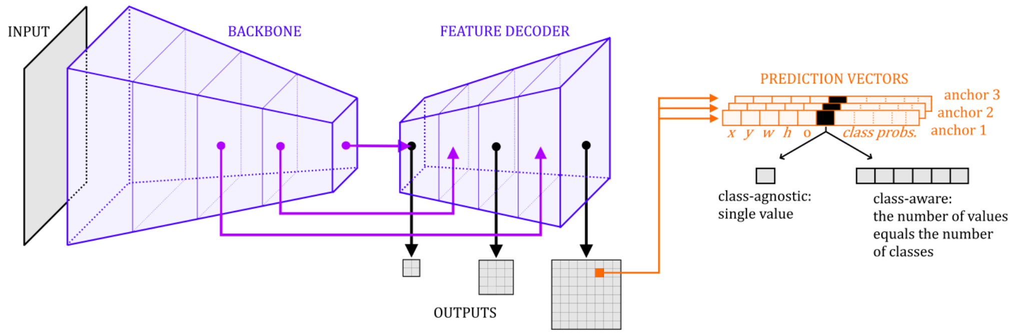

Figure 1. The figure illustrates how the YOLO architecture is modified to estimate the distance of objects. The part enriching YOLO by distance estimation is visualized by the black box in the prediction vector (marked by orange color).

2.3. Updating the YOLOv3 Loss Function

Dist-YOLO extends the original YOLOv3 loss function ℓ of a certain scale into the form [31]:

$$\ell = \sum_{i=0}^{G^w G^h} \sum_{j=0}^{n^a} q_{i,j} \Big[\ell_1(i,j) + \ell_2(i,j) + \ell_3(i,j) + \ell_5(i,j) \Big] + \ell_4(i,j),$$

where $\ell_1(i,j)$ is a loss of bounding box center prediction, $\ell_2(i,j)$ is a box dimensions loss, $\ell_3(i,j)$ is the confidence loss, $\ell_4(i,j)$ is the class prediction loss, and $\ell_5(i,j)$ is the distance loss. Finally, $q_{i,j} \in \{0,1\}$ is a constant indicating whether the i-th cell and the j-th anchor contains an object or not. The loss iterates over $G^w G^h$ grid cells and n^a anchors. The parts $\ell_1, \ldots, \ell_4$ are taken from YOLOv3. Part ℓ_5 is new and extends YOLOv3 with the functionality of distance estimation. In the following formulas, we use $\hat{\cdot}$ to denote predictions of the network. The parts of the loss function are, according to [31], defined as follows:

$$\ell_1(i,j) = z_{i,j} \Big[H(c^x_{i,j}, \widehat{c}^x_{i,j}) + H(c^y_{i,j}, \widehat{c}^y_{i,j}) \Big],$$

where $c^x_{i,j}$ and $c^y_{i,j}$ are coordinates of the center of a box, $H(\cdot,\cdot)$ is the binary cross-entropy, and $z_{i,j} = 2 - w_{i,j} h_{i,j}$ serves for a relative weighting of (i,j)-th box size according to its width $w_{i,j}$ and height $h_{i,j}$.

$$\ell_2(i,j) = 0.5 z_{i,j} \left[\left(log\left(\frac{w_{i,j}}{a^w_j} \right) - \hat{w}_{i,j} \right)^2 + \left(log\left(\frac{h_{i,j}}{a^h_j} \right) - \hat{h}_{i,j} \right)^2 \right],$$

where a^w_j and a^h_j are the width and height of the j-th anchor.

$$\ell_3(i,j) = q_{i,j} H(q_{i,j}, \hat{q}_{i,j}) + (1 - q_{i,j}) H(q_{i,j}, \hat{q}_{i,j}) I_{i,j},$$

where $\hat{q}_{i,j}$ is the predicted confidence, and $I_{i,j}$ is a mask, which excludes the part of a loss for the i-th cell if $q_{i,j} = 0$, but its prediction has IoU > 0.5.

$$\ell_4(i,j) = \sum_{k=0}^{c} H(C_{i,j,k} - \hat{C}_{i,j,k}),$$

where $C^{i,j,k}$ is k-th class probability in i-th cell. The distance loss is defined as

$$\ell_5(i,j) = \omega(\hat{d}_{i,j} - d_{i,j})^2$$

for the class-agnostic version and

$$\ell_5(i,j) = \omega \sum_{k=0}^{c} C_{i,j,k}(\hat{d}_{i,j,k} - d_{i,j,k})^2$$

for the class-aware version, where ω is a weighting constant preventing the distance loss being preferable than the other losses. In our experiment, we set $\omega = 1 \times 10^{-2}$.

2.4. Updating YOLOv3 Training Data

The distances enriching the data can be taken using lidar information in datasets such as KITTI [14], Waymo Open Dataset [44], Berkeley DeepDrive Dataset [45], or nuScenes [46], etc. Adding distance into the prediction vector means that it uses the same features as the bounding box regressor, and the features can also be trained to minimize the loss of distance estimation. That is the difference from other modifications of YOLO, which build distance estimation on predictions taken from the already trained model.

3. Benchmark and Results

We conducted the experiment on the KITTI dataset [14], namely the KITTI 3D Object Detection Evaluation 2017, which contains 7481 training and 7518 test images. Because groundtruth labels are available only for training data, we split the original training set into the training part consisting of 5241 images and the testing part consisting of 2240 images, converted to a fixed resolution of 1216 × 366 pixels. The categories were pedestrian, car, van, truck, sitting person, cyclist, and tram. Each label contained information about its location in the image, distance, and rotation with respect to the camera. We used only the first two pieces of information. The distance was expressed in meters. The lowest distances were in the negative range for objects passing the camera, and we clipped such distances to zero. The largest distances rarely exceeded 150 m; we clipped them into 150 max. Only a few objects exceeded the distance of 90 m. The distances were normalized into $[0, 1]$ for training. The distribution of the distances is shown in Figure 2. We measured the correlation between the object's height and the observed distance for the class car in the testing dataset. It was -0.74, which means the distance could not be precisely estimated only using the object's size. That is why approaches using size information only, such as DisNet [2], cannot reach state-of-the-art results.

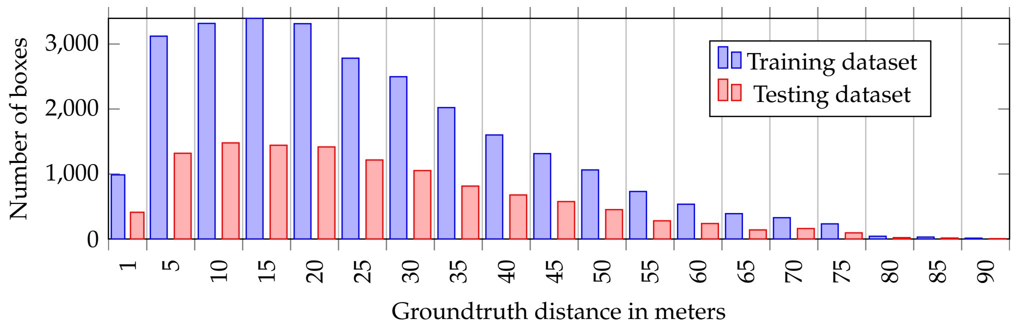

Figure 2. The distribution of object distances in training and testing datasets. The graph is clipped to the maximum distance of 90 m to increase visibility.

In the benchmark, all models were trained using the following settings: The input of YOLO was a downscaled image with the resolution of 608 × 192 pixels (used for both training/testing scenarios), the optimizer was Adam with $\alpha = 1 \times 10^{-3}$ and 'reduce learning rate on a plateau' functionality with patience 10 and factor 0.5. The data were online modified by augmentation consisting of padding, scaling, flipping, and brightness/contrast/color adjustments to prevent overfitting. The batch size was set to 24. The training ran for 100 epochs and was evaluated on a validation dataset after each epoch; the best model was saved. The evaluation dataset was 10% of training samples. Finally, the best model was evaluated on the test set.

The text below marks class-agnostic distance YOLO as 'Dist-YOLOv3 G' and class-aware distance YOLO as 'Dist-YOLOv3 W'.

3.1. Evaluation Criteria

For the evaluation of the box detection ability, the mAP (mean average precision) index following the COCO standard [47,48] was used.

Regarding the quality of distance estimation, we defined ε_A and ε_R metrics expressing the mean absolute and relative distance estimation error. The mean absolute distance error (MAE) is defined as

$$\varepsilon_A = \frac{1}{n}\sum_{i=1}^{n} |d_i - \hat{d}_i|,$$

and the mean relative distance error as

$$\varepsilon_R = \frac{1}{n}\sum_{i=1}^{n} \frac{|d_i - \hat{d}_i|}{\max(d_i, 1)},$$

where n is the number of found bounding boxes. Further, d, $\hat{d}$ represents the paired vectors of the groundtruth and predictions, respectively.

3.2. Preserving YOLO Performance

The first experiment was aimed to verify our motivation, namely, the assumption that the network could use the same inner features for BB coordinates and distance estimation, leading to synergy. To verify it, we trained three models: the original YOLOv3, Dist-YOLOv3 with class-agnostic distance, and Dist-YOLOv3 with class-aware distance. All three models were trained with the same setting and had the same backbone.

The numerical results are presented in Table 1 and supported our assumptions. For both types of Dist-YOLOv3, the mAP value increased. Focusing on mAP$_{.5:.95}$, the class-agnostic Dist-YOLOv3 yielded a performance 108% of the original YOLOv3 and class-aware Dist-YOLOv3 yielded 118%. The justification lies in a technique called *Auxiliary task*. It has been proven [49,50] that the correct choice of an appropriate auxiliary task may lead to the performance improvement of the primary task.

Table 1. Measurement of bounding box precision detection. 'YOLOv3' represents result for the original YOLOv3 implementation. 'Dist-YOLOv3 G' is the result for our class-agnostic approach, and 'Dist-YOLOv3 W' is the result for our class-aware approach.

Model	**Params**	**mAP$_{.5}$**	**mAP$_{.5:.95}$**
YOLOv3	42.56 M	74.3	28.3
Dist-YOLOv3 G	42.57 M	76.2	30.7
Dist-YOLOv3 W	42.60 M	77.1	33.5

3.3. Measure the Error of Distance Estimation

We have proven Dist-YOLO outperforms the standard YOLO regarding the accuracy of box detection. The second experiment validated the ability of Dist-YOLO to estimate the objects' distance precisely. The distance was evaluated using the defined ε_A and ε_R metrics. The metrics could be evaluated only for detected boxes; it was pointless to evaluate the distance of an object that was not detected. To distinguish what was detected, we used the IoU threshold of 0.5; objects detected with IoU lower than the threshold were discarded.

The results for Dist-YOLOv3 are shown in Table 2. We express the number of detected boxes for a certain class, the minimum and maximum distance error, and the mean distance error for completeness. The mean error was not computed from absolute values; therefore, it should be zero when no bias in the predictions is included. Contrary to the previous experiment, class-agnostic Dist-YOLO yielded better results than class-aware, which can even be marked as not working adequately due to the high values of ε_A, ε_R and high positive bias.

Table 2. Measurement of distance error in meters for Dist-YOLOv3 G/W. The values in columns Min, Mean, and Max, ε_A are expressed in meters. The value of column ε_R is relative, where $\varepsilon_R = 1$ equals to 100%.

Class	**Dist-YOLOv3 G**						**Dist-YOLOv3 W**					
	#	**Min**	**Mean**	**Max**	ε_A	ε_R	**#**	**Min**	**Mean**	**Max**	ε_A	ε_R
Pedestrian	1150	−16.81	−0.54	7.73	1.75	0.11	1117	−8.93	3.83	39.48	4.87	0.41
Car	7682	−24.87	−0.81	14.47	2.49	0.11	7798	−13.24	8.75	42.10	10.68	0.44
Van	749	−20.49	0.46	18.00	3.55	0.15	751	−26.88	13.94	42.25	14.17	0.64
Truck	318	−34.36	1.16	15.93	4.63	0.14	319	−27.33	4.51	26.30	6.49	0.21
Person sitting	57	−7.09	−0.59	4.74	1.84	0.22	56	−2.27	4.35	16.15	4.64	0.64
Cyclist	406	−17.79	−1.25	14.66	2.37	0.11	409	−4.16	14.32	46.80	14.38	0.85
Tram	135	−26.66	1.85	19.57	4.72	0.19	132	−22.37	−2.65	11.36	4.41	0.17
All	10,497	−34.36	−0.61	19.57	2.57	0.11	10,582	−27.33	8.52	46.80	10.22	0.46

3.4. Comparison with the Other Methods

A direct comparison is not straightforward as there is no standard, and different authors report their results using various settings; therefore, the following numbers are mainly illustrative. Zhu et al. [18] achieved $\varepsilon_R = 0.25$ on their KITTI validation dataset. Ali et al. [51] tested only nonoverlapping cars with the maximum distance of 70 m and achieved $\varepsilon_R = 0.29$; they also reported that DORN [52] yielded $\varepsilon_R = 0.11$ using the same setting. Mauri et al. [26] evaluated KITTI for three classes: car, person, and cyclist. Considering the test scenario, the achieved ε_R for these three classes was 0.16, 0.42, and 1.04, which shows the reasonableness of our scheme.

3.5. Ablation Study

Firstly, we investigated the dependency of the distance error on the object's real distance. Because class-aware Dist-YOLO showed a poor result in the previous experiment, we considered only class-agnostic Dist-YOLO. The absolute and relative distance errors are visualized in Figures 3 and 4, respectively. Concerning Figure 3, we can claim that the absolute error grows according to a polynomial curve until distances higher than 80, where the trend is broken. The reason is that few objects lie in this interval and, therefore, the measured value was not computed from a statistically significant number of samples. Similar behavior is also reflected in Figure 4. Here, we can see that the relative error was almost constant for all distances except one meter. That is caused by the fact that even a 30 cm distance error on such a distance yields an error of 0.3, which makes the task extremely difficult and is probably also affected by the inaccuracy of the distance capturing process itself.

The next investigated aspect was the correlation between variables. Firstly, we measured the correlation between the quality of bounding box detection and the distance error. It was −0.27 for IoU and ε_A and 0.05 for IoU and ε_R. Secondly, we measured that the correlation between the height of an object and ε_A was −0.55, i.e., the lower the box is, the larger the absolute error produced, which is consistent with the values in Figure 3. For height and ε_R, it was 0.11, which is again consistent with the values in Figure 4; the positive correlation is caused by the large error for close boxes. Note, all of these correlations are computed for class 'car', and IoU > 0.5 between the predictions and labels.

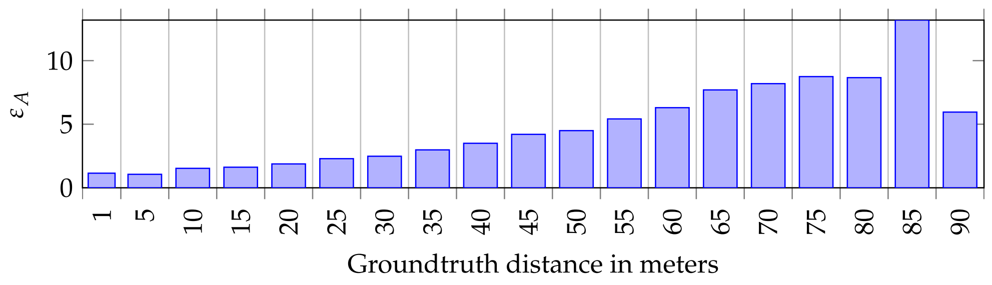

Figure 3. Dependency of ε_A on the object's distance.

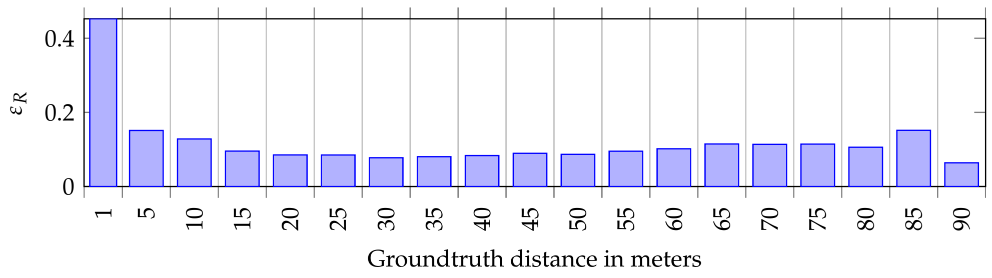

Figure 4. Dependency of ε_R on the object's distance.

The last aspect we evaluated was the computation speed. We measured the time on an RTX2060 SUPER graphics card, where the time represents the duration of executing the *predict()* method with a batch size equal to one and without further optimization such as conversion into TensorRT. The prediction time was equal for all three models, 22 ms per image, i.e., it ran approx 45 frames per second. The equal processing time confirmed that the increase in parameters in the output prediction vectors was negligible compared to the complexity of the rest of the model.

3.6. Examples

Finally , examples of detections are provided in Figure 5. We present four scenes with objects representing various classes and distances. Every detected object is encapsulated in the bounding box. The color and the title of the box refer to the detected class. The two numbers in the label refer to the distance of the object-the first value represents the detected distance, the one after the slash represents the ground-truth distance. Both values are in meters.

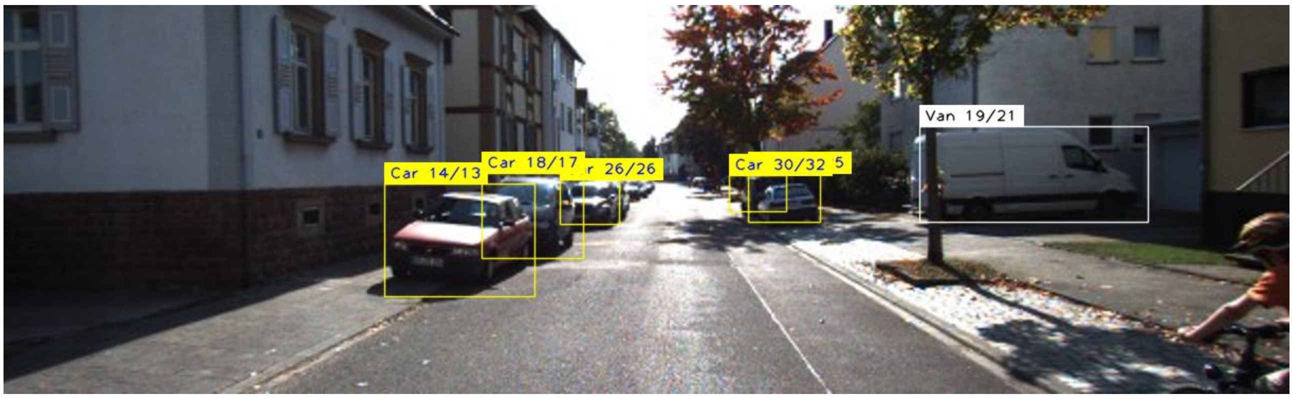

Figure 5. *Cont.*

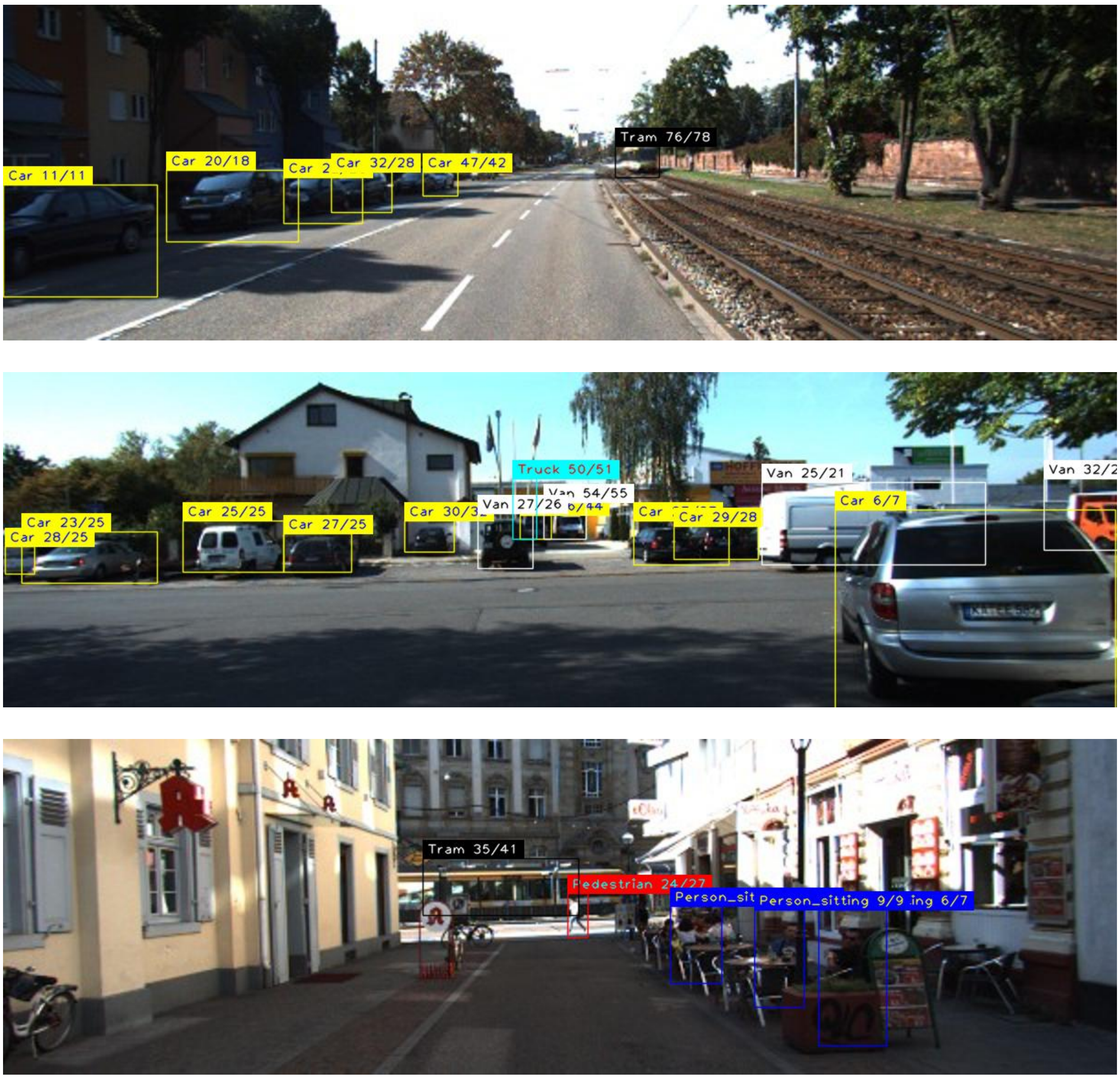

Figure 5. The figure shows bounding box prediction and distance estimation on the test set using Dist-YOLO. The distance (in meters) is in the format $\hat{d}/d$, where $\hat{d}$ is the prediction itself, and d represents the groundtruth label.

4. Discussion, Open Issues, and Future Work

Training with distance estimation limits used data augmentation techniques. Spatial transformations such as mosaicing [5] or strong resize can distort the visual size of objects and confuse the network. On the other hand, reducing augmentation can increase overfitting. Therefore, the suitable way to train Dist-YOLO is to use curriculum learning [53]. Firstly, the model was trained with data augmentation and we froze (set to zero) the part of the loss responsible for distance estimation to train bounding box detection only. Then, the complete loss was used, and only the not-distort-object-size augmentations were used.

From Figure 4, it is obvious that the percentage relative error of the distance was for most cases below 10%. The outlier was the distance of one meter, where the relative error was 45%. That behavior is caused by the current form of the loss function, where the squared distance is computed. We experimented with the relative distance loss function, which would force the network to predict a distance with similar relative error for all distances, including one meter. Such a loss function for the class-agnostic version is:

$$\ell_5(i,j) = \omega \frac{|d_{i,j} - \hat{d}_{i,j}| + \sigma}{d_{i,j} + \sigma},$$

where $\sigma \in [0,1]$ is a positive smoothing factor to avoid instability for the cases when $d_{i,j} \approx 0$. However, we were unable to train the model with this loss function due to high loss, even for bounding boxes.

Because the new versions of YOLO, namely v4 [5] and v5, use the same principles as YOLOv3, the proposed scheme of monocular distance estimation can be integrated into these new variants. This realization remains a future work.

5. Summary

We started with YOLOv3, a fast one-stage object detector. In the literature overview, we showed that there were approaches modifying YOLOv3 to monocular absolute distance estimation, but no one fully integrated the distance estimation functionality into YOLO's architecture. To achieve that, we extended the output prediction vector, modified the loss function by the part responsible for absolute distance estimation, and described how to modify the training data. In the benchmark, we verified that distance estimation was complementary to bounding box estimation and, therefore, increasds the bounding box detection precision compared to the standard YOLOv3. We evaluated two versions of the designed Dist-YOLO, namely class-agnostic and class-aware, with the finding that only class-agnostic yielded satisfactory results. Benchmarking the KITTI dataset, the mean absolute distance error was 2.5 m, and the mean relative error was 11%, which support the claim that the full integration of distance estimation into YOLO's functionality can reach a significantly better result compared to solutions that build distance estimation on the top of the model's output. Finally, we reported that the computation speed was identical for both YOLOv3 and Dist-YOLOv3. The implementation is available at https://gitlab.com/EnginCZ/yolo-with-distance (accessed on 8 October 2021).

Author Contributions: Conceptualization, M.V. and P.H.; Data curation, T.N.; formal analysis, M.V. and P.H. and T.N.; methodology, M.V. and T.N.; software, M.V.; validation, M.V. and P.H. and T.N.; visualisation, M.V.; writing, M.V. and P.H. All authors have read and agreed to the published version of the manuscript.

Funding: The work was supported by ERDF/ESF "Centre for the development of Artificial Intelligence Methods for the Automotive Industry of the region" (No. CZ.02.1.01/0.0/0.0/17_049/0008414).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. These data can be found here: http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d (accessed on 14 August 2021). The source codes implementing the presented approach are freely available at: https://gitlab.com/EnginCZ/yolo-with-distance (accessed 14 August 2021).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

YOLO	You Only Look Once
mAP	Mean Average Precision
MAE	Mean Average (Distance) Error
BB	Bounding Box
IoU	Intersection over Union

References

1. Rukhovich, D.; Mouritzen, D.; Kaestner, R.; Rufli, M.; Velizhev, A. Estimation of Absolute Scale in Monocular SLAM Using Synthetic Data. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), Seoul, Korea, 27–28 October 2019, pp. 803–812. [CrossRef]
2. Haseeb, M.A.; Guan, J.; Ristic-Durrant, D.; Gräser, A. DisNet: A novel method for distance estimation from monocular camera. In Proceedings of the 10th Planning, Perception and Navigation for Intelligent Vehicles (PPNIV18), Madrid, Spain, 1 October 2018.
3. Zhang, K.; Xie, J.; Snavely, N.; Chen, Q. Depth sensing beyond lidar range. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 1692–1700. [CrossRef]
4. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
5. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
6. Jocher, G.; Stoken, A.; Borovec, J.; NanoCode012; ChristopherSTAN; Liu, C.; Laughing; Tkianai; Hogan, A.; Lorenzomammana; et al. Ultralytics/yolov5: v3.1-Bug Fixes and Performance Improvements. 2020. Available online: https://zenodo.org/record/4154370#.YfJaRfgRXQw (accessed on 26 January 2022).
7. Van Breugel, F.; Morgansen, K.; Dickinson, M.H. Monocular distance estimation from optic flow during active landing maneuvers. *Bioinspir. Biomimetics* **2014**, *9*, 025002. [CrossRef] [PubMed]
8. Ranftl, R.; Bochkovskiy, A.; Koltun, V. Vision Transformers for Dense Prediction. *arXiv* **2021**, arXiv:2103.13413.
9. Luo, X.; Huang, J.B.; Szeliski, R.; Matzen, K.; Kopf, J. Consistent video depth estimation. *ACM Trans. Graph. (TOG)* **2020**, *39*, 71–1. [CrossRef]
10. Kumari, S.; Jha, R.R.; Bhavsar, A.; Nigam, A. Autodepth: Single image depth map estimation via residual cnn encoder-decoder and stacked hourglass. In Proceedings of the 2019 IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 22–25 September 2019; pp. 340–344. [CrossRef]
11. Park, M.G.; Yoon, K.J. As-planar-as-possible depth map estimation. *Comput. Vis. Image Underst.* **2019**, *181*, 50–59. [CrossRef]
12. Zhou, Y.; Tuzel, O. Voxelnet: End-to-end learning for point cloud based 3d object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4490–4499. [CrossRef]
13. Lehner, J.; Mitterecker, A.; Adler, T.; Hofmarcher, M.; Nessler, B.; Hochreiter, S. Patch Refinement–Localized 3D Object Detection. *arXiv* **2019**, arXiv:1910.04093.
14. Geiger, A.; Lenz, P.; Urtasun, R. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, USA , 16–21 June 2012.
15. Geiger, A.; Lenz, P.; Urtasun, R. KITTI 3D Object Detection Benchmark. 2012. Available online: http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d (accessed on 14 July 2019).
16. Saxena, A.; Chung, S.H.; Ng, A.Y. Learning depth from single monocular images. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2006; pp. 1161–1168.
17. Garg, R.; BG, V.K.; Carneiro, G.; Reid, I. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 740–756.
18. Zhu, J.; Fang, Y. Learning object-specific distance from a monocular image. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 3839–3848.
19. Hu, H.N.; Cai, Q.Z.; Wang, D.; Lin, J.; Sun, M.; Krahenbuhl, P.; Darrell, T.; Yu, F. Joint monocular 3D vehicle detection and tracking. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 5390–5399.
20. Huang, L.; Zhe, T.; Wu, J.; Wu, Q.; Pei, C.; Chen, D. Robust inter-vehicle distance estimation method based on monocular vision. *IEEE Access* **2019**, *7*, 46059–46070. [CrossRef]
21. Natanael, G.; Zet, C.; Foşalău, C. Estimating the distance to an object based on image processing. In Proceedings of the 2018 International Conference and Exposition on Electrical Furthermore, Power Engineering (EPE), Iasi, Romania, 18–19 October 2018; pp. 0211–0216.
22. Haseeb, M.A.; Ristić-Durrant, D.; Gräser, A. Long-range obstacle detection from a monocular camera. In Proceedings of the ACM Computer Science in Cars Symposium (CSCS), Munich, Germany, 13–14 September 2018.
23. Chen, Z.; Khemmar, R.; Decoux, B.; Atahouet, A.; Ertaud, J.Y. Real time object detection, tracking, and distance and motion estimation based on deep learning: Application to smart mobility. In Proceedings of the 2019 Eighth International Conference on Emerging Security Technologies (EST), Colchester, UK, 22–24 July 2019, pp. 1–6.
24. Godard, C.; Mac Aodha, O.; Brostow, G.J. Unsupervised monocular depth estimation with left-right consistency. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017, pp. 270–279.
25. Strbac, B.; Gostovic, M.; Lukac, Z.; Samardzija, D. YOLO Multi-Camera Object Detection and Distance Estimation. In Proceedings of the 2020 Zooming Innovation in Consumer Technologies Conference (ZINC), Novi Sad, Serbia, 26–27 May 2020; pp. 26–30.
26. Mauri, A.; Khemmar, R.; Decoux, B.; Haddad, M.; Boutteau, R. Real-time 3D multi-object detection and localization based on deep learning for road and railway smart mobility. *J. Imaging* **2021**, *7*, 145. [CrossRef]
27. You, Y.; Wang, Y.; Chao, W.L.; Garg, D.; Pleiss, G.; Hariharan, B.; Campbell, M.; Weinberger, K.Q. Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving. *arXiv* **2019**, arXiv:1906.06310.
28. 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.

29. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
30. Ge, Z.; Liu, S.; Wang, F.; Li, Z.; Sun, J. Yolox: Exceeding yolo series in 2021. *arXiv* **2021**, arXiv:2107.08430.
31. Hurtik, P.; Molek, V.; Hula, J.; Vajgl, M.; Vlasanek, P.; Nejezchleba, T. Poly-YOLO: Higher speed, more precise detection and instance segmentation for YOLOv3. *arXiv* **2020**, arXiv:2005.13243.
32. Bolya, D.; Zhou, C.; Xiao, F.; Lee, Y.J. YOLACT: Real-time Instance Segmentation. *arXiv* **2019**, arXiv:1904.02689.
33. Bolya, D.; Zhou, C.; Xiao, F.; Lee, Y.J. YOLACT++: Better Real-time Instance Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *44*, 1108–1121. [CrossRef]
34. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef]
35. Yu, J.; Zhang, W. Face mask wearing detection algorithm based on improved YOLO-v4. *Sensors* **2021**, *21*, 3263. [CrossRef]
36. Roy, A.M.; Bose, R.; Bhaduri, J. A fast accurate fine-grain object detection model based on YOLOv4 deep neural network. *Neural Comput. Appl.* **2022**, 1–27. [CrossRef]
37. Yi, Z.; Shen, Y.; Zhang, J. An improved tiny-yolov3 pedestrian detection algorithm. *Optik* **2019**, *183*, 17–23. [CrossRef]
38. Jiao, Z.; Zhang, Y.; Xin, J.; Mu, L.; Yi, Y.; Liu, H.; Liu, D. A deep learning based forest fire detection approach using UAV and YOLOv3. In Proceedings of the 2019 1st International Conference on Industrial Artificial Intelligence (IAI), Shenyang, China, 23–27 July 2019; pp. 1–5.
39. Zhou, F.; Zhao, H.; Nie, Z. Safety Helmet Detection Based on YOLOv5. In Proceedings of the 2021 IEEE International Conference on Power Electronics, Computer Applications (ICPECA), Shenyang, China, 22–24 January 2021; pp. 6–11.
40. De Guzman, S.R.C.; Tan, L.C.; Villaverde, J.F. Social Distancing Violation Monitoring Using YOLO for Human Detection. In Proceedings of the 2021 IEEE 7th International Conference on Control Science and Systems Engineering (ICCSSE), Qingdao, China, 30 July–1 August 2021; pp. 216–222. [CrossRef]
41. Rahim, A.; Maqbool, A.; Rana, T. Monitoring social distancing under various low light conditions with deep learning and a single motionless time of flight camera. *PLoS ONE* **2022**, *16*. [CrossRef]
42. Wu, P.; Li, H.; Zeng, N.; Li, F. FMD-Yolo: An efficient face mask detection method for COVID-19 prevention and control in public. *Image Vis. Comput.* **2022**, *117*, 104341. [CrossRef]
43. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference of International Conference on Machine Learning, Lille, France, 7–9 July 2015; Volume 37, pp. 448–456.
44. Qi, C.R.; Zhou, Y.; Najibi, M.; Sun, P.; Vo, K.; Deng, B.; Anguelov, D. Offboard 3D Object Detection from Point Cloud Sequences. *arXiv* **2021**, arXiv:2103.05073.
45. Yu, F.; Xian, W.; Chen, Y.; Liu, F.; Liao, M.; Madhavan, V.; Darrell, T. Bdd100k: A diverse driving video database with scalable annotation tooling. *arXiv* **2018**, arXiv:1805.04687.
46. Caesar, H.; Bankiti, V.; Lang, A.H.; Vora, S.; Liong, V.E.; Xu, Q.; Krishnan, A.; Pan, Y.; Baldan, G.; Beijbom, O. nuScenes: A multimodal dataset for autonomous driving. *arXiv* **2019**, arXiv:1903.11027.
47. Lin, T.Y.; Maire, M.; Belongie, S.; Bourdev, L.; Girshick, R.; Hays, J.; Perona, P.; Ramanan, D.; Zitnick, C.L.; Dollár, P. Microsoft COCO: Common Objects in Context. *arXiv* **2015**, arXiv:1405.0312.
48. Lin, T.Y.; Maire, M.; Belongie, S.; Bourdev, L.; Girshick, R.; Hays, J.; Perona, P.; Ramanan, D.; Zitnick, C.L.; Dollár, P. Microsoft COCO—Common Objects in Context. 2015. Available online: https://cocodataset.org//#detection-eval (accessed on 15 October 2021).
49. Mordan, T.; Thome, N.; Henaff, G.; Cord, M. Revisiting multi-task learning with rock: A deep residual auxiliary block for visual detection. In Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS), Montreal, QC, Canada, 2–8 December 2018.
50. Castellano, G.; Castiello, C.; Mencar, C.; Vessio, G. Crowd detection in aerial images using spatial graphs and fully-convolutional neural networks. *IEEE Access* **2020**, *8*, 64534–64544. [CrossRef]
51. Ali, A.; Hassan, A.; Ali, A.R.; Khan, H.U.; Kazmi, W.; Zaheer, A. Real-time vehicle distance estimation using single view geometry. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Snowmass, CO, USA, 1–5 March 2020; pp. 1111–1120.
52. Fu, H.; Gong, M.; Wang, C.; Batmanghelich, K.; Tao, D. Deep ordinal regression network for monocular depth estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 2002–2011.
53. Bengio, Y.; Louradour, J.; Collobert, R.; Weston, J. Curriculum learning. In Proceedings of the 26th Annual International Conference on Machine Learning, Montreal, QC, Canada, 14–18 June 2009; pp. 41–48. [CrossRef]