

Article

ConvGRU Hybrid Model Based on Neural Ordinary Differential Equations for Continuous Dynamics Video Object Detection

Linbo Qian ^{1,2}, Shanlin Sun ^{2,3,*} and Shike Long ^{2,3} 

¹ School of Computer Electronics and Information, Guangxi University, Nanning 530004, China; lbqian93@st.gxu.edu.cn

² Guangxi Colleges and Universities Key Laboratory of Unmanned Aerial Vehicle System and Technology Application, Guilin University of Aerospace Technology, Guilin 541004, China; 2018087@guat.edu.cn

³ School of Aeronautics and Astronautics, Guilin University of Aerospace Technology, Guilin 541004, China

* Correspondence: shy1997@126.com

Abstract: Video object detection involves identifying and localizing objects within video frames over time. However, challenges such as real-time processing requirements, motion blur, and the need for temporal consistency in video data make this task particularly demanding. This study proposes a novel hybrid model that integrates Neural Ordinary Differential Equations (Neural ODEs) with Convolutional Gated Recurrent Units (ConvGRU) to achieve continuous dynamics in object detection for video data. First, it leverages the continuous dynamics of Neural ODEs to define the hidden state transitions between observation points, enabling the model to naturally align with real-world time-based processes. Second, we present the FPN-Up module, which combines high-level semantic information with low-level spatial details to enhance the exploitation of multi-layer feature representations. Finally, we integrate a CBAM attention module into the detection head, enabling the model to emphasize the most salient input feature regions, thereby elevating detection precision while preserving the existing network structure. Evaluation on the KITTI object detection dataset reveals that our proposed model outperforms a vanilla video object detector by 2.8% in mAP while maintaining real-time processing capabilities.



Academic Editors: Yang Xu and Delin Luo

Received: 27 March 2025

Revised: 10 May 2025

Accepted: 12 May 2025

Published: 16 May 2025

Citation: Qian, L.; Sun, S.; Long, S. ConvGRU Hybrid Model Based on Neural Ordinary Differential Equations for Continuous Dynamics Video Object Detection. *Electronics* **2025**, *14*, 2033. <https://doi.org/10.3390/electronics14102033>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: video object detection; neural ODE; attention mechanism; feature fusion

1. Introduction

Object detection, a pivotal area in deep learning research, underpins numerous applications and drives interdisciplinary advancements. This computer vision task merges classification, which determines the existence of objects, with regression, which estimates their spatial positions, to recognize and pinpoint objects within images or video streams. Recent advancements in static image object detection [1–6] have significantly improved both accuracy and inference speed. Despite these advancements, the closely related field of video object detection remains comparatively underexplored, yet it holds critical importance for real-world applications. Video object detection extends the principles of static image object detection to time-based data, incorporating temporal information that captures object motion, appearance changes, and interactions over time. The temporal dimension in video data is crucial for applications such as UAV navigation, target tracking, encircling, and autonomous driving, where object detection relies on processing dynamic, sequential data [7–10].

Leveraging temporal information in video data can significantly improve the performance of object detection algorithms. However, video-based object detection also intro-

duces unique challenges, including motion blur, strict real-time processing requirements, and variability in frame rates. Addressing these challenges is essential for developing robust and efficient video object detection systems capable of performing reliably across diverse real-world scenarios [11].

Video object detection builds upon static image object detection. Modern deep learning-based frame-wise object detection models predominantly rely on convolutional neural network (CNN) architectures. These models can be broadly classified into two types: two-stage and single-stage detectors. Two-stage detectors, predominantly the R-CNN family [3,12–15], function by initially producing region proposals, which are then categorized into distinct object classes. Due to their two-stage nature, these methods often involve complex loss functions and non-differentiable components, which complicate end-to-end training. In contrast, single-stage detectors, such as the You Only Look Once (YOLO) series [1,16,17] and the Single-Shot Multibox Detector (SSD) [2], employ a unified pipeline to directly predict object locations and categories in a single forward pass. This streamlined approach enables all computations to be performed in one step, making these detectors significantly faster and more efficient than their two-stage counterparts. Furthermore, because single-stage detectors operate as a unified framework, they facilitate end-to-end training, simplifying the overall training process. Nowadays, many applications still rely on static object detection algorithms for video-based data. While this approach can achieve real-time processing in certain scenarios, it overlooks the temporal context inherent in video data. This omission can lead to challenges when detecting objects across consecutive frames, particularly for moving objects, where temporal coherence is critical.

Various approaches have been proposed to utilize the additional temporal information from video data. One of the earliest methods involves post-processing the results obtained from single-image object detection [18,19]. And there are ways of using a tracking model to optimize the detection results [20]. Additionally, feature filtering techniques have been developed to extract representative features from a frame, prioritizing key regions for detection and guiding the selection of relevant features in subsequent frames [7,21]. Another category of methods directly incorporates motion and temporal information into the network architecture. For instance, optical flow has been widely used to capture pixel-wise motion between consecutive frames, enhancing the extraction of temporal context [22]. However, this approach is limited to pairwise frame analysis and cannot fully exploit the temporal dimension across longer sequences. Zhou et al. [23] and He et al. [24] proposed to use Spatial–Temporal transformers to achieve object detection. Cui et al. [25] proposed to fuse temporal feature from adjacent frames to produce stronger features for detection tasks. Xu et al. [26] proposed to aggregate spatial–temporal information to enhance video object detection. Alternatively, recurrent neural network (RNN)-based architectures, such as Convolutional LSTM and Convolutional GRU, have been employed to model temporal dependencies more effectively [27–29]. These networks can learn spatiotemporal features over a sequence of frames, generating richer representations that integrate both spatial and temporal information for the detection task. In this work, we adopt an RNN-like architecture to perform video object detection.

RNNs are commonly employed to process temporal features in video data, for the overall process resembles a dynamical system; in this work, we sometimes use the word “dynamics” to describe the hidden state of a temporal process. There exists two predominant approaches to model the dynamics of video data. Both methods utilize recurrent units to store and extract temporal information, but they differ in their input representations. The first approach is the box-level methods [27,30,31]. In this method, sequential input frames are initially processed using a standard static image detection model, such as YOLO or SSD. The resulting bounding boxes are then fed into a recurrent unit, which aggregates temporal

information across frames based on these bounding box predictions. The recurrent unit refines the bounding box information to produce the final detection results, including object classes and locations. The second approach is the feature-based method [28,29,32]. Here, a feature extractor is used to generate feature maps from the input frames, then these feature maps are sequentially passed into a recurrent unit, which captures temporal dependencies across the video sequence. Finally, the features contain both spatial and temporal information are forwarded to a detection head, which outputs the predictions for object classes and locations.

From previous works, it has been demonstrated that fusing temporal and spatial features using recurrent units can enhance model performance [28–30]. However, whether such methods fully capture the environmental context remains an open question. For example, when using traditional recurrent methods to process temporal features from video data, the input typically consists of discrete frames sampled from a continuous time-based process. This introduces problems, such as non-uniform frame intervals, which are often assumed to be uniform in an RNN architecture. Additionally, the model does not inherently recognize that it is processing a continuous-time process, as it operates on discrete inputs. These limitations raise concerns about whether the design architecture of recurrent units is well suited for real-world video data.

Building on the aforementioned challenges, in this work, we propose a NODE-ConvGRU hybrid model for video object detection, using the Recurrent Multi-frame Single Shot Detector (RMf-SSD) [28] as a baseline. Our approach introduces several key contributions. First, we use Neural ODE to define the hidden states between observations, making our model better align with real-world time-based processes. And this modification allows the model to explicitly account for the time intervals between consecutive observations, addressing one of the limitations of traditional recurrent architectures. Second, we propose a FPN-Up (Feature Fusion Network–Up) feature fusion module to effectively leverage hierarchical features extracted by CNNs. The enhancement facilitates multi-scale feature extraction, enhancing the model’s capacity to accurately detect objects across diverse sizes. Finally, we integrate the Convolutional Block Attention Module (CBAM) [33] into the model’s detection head to enhance feature representation, boosting overall performance by emphasizing critical spatial and channel-wise features.

2. Related Work

2.1. Static Image Object Detection

Due to their superior feature extraction and adaptive learning capabilities compared to traditional approaches like hand-crafted features [34], modern object detection predominantly relies on CNN-based methods. These detectors can be broadly categorized into two groups: two-stage detectors and one-stage detectors.

Two-stage detectors [3,12–15] operate by first generating region proposals, which are then classified using CNNs. In contrast, single-stage detectors [1,16,17] make predictions in a single forward pass, achieving faster inference speeds ideal for real-time applications. Due to their architecture, a two-stage approach is generally slower and more complex compared to single-stage detectors.

Here, we introduce SqueezeDet [4], a single-stage, fully convolutional detector built on SqueezeNet [35]. Engineered to significantly reduce the parameter count while retaining accuracy comparable to AlexNet, the SqueezeNet architecture empowers SqueezeDet to deliver exceptional computational efficiency without sacrificing detection performance. This makes it particularly well suited for video object detection tasks, where efficiency is of utmost importance.

SqueezeNet minimizes the number of parameters in convolutional layers by utilizing a structure called the Fire module. As depicted in Figure 1, the Fire module consists of two sequential components: a squeeze layer and an expand layer. The squeeze layer exclusively employs 1×1 convolutional filters, functioning as a bottleneck to diminish the dimensionality of the input prior to the application of computationally intensive 3×3 convolutions, which can significantly reduce the number of parameters while still being able to preserve the essential information. The expand layer, on the other hand, incorporates both 1×1 and 3×3 convolutional filters, enabling the capture of low-level features through the former and higher-level spatial features through the latter. And by ensuring that the number of 1×1 filters in the squeeze layer is smaller than the total number of filters in the expand layer, the architecture effectively limits the number of input channels to the 3×3 filters, thereby further reducing the overall model parameters.

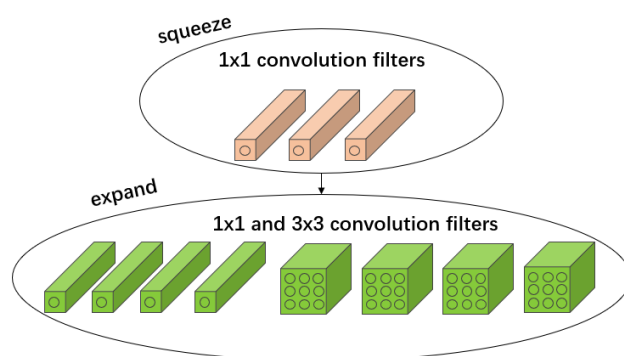


Figure 1. Fire module. Squeezing channels with 1×1 filters, while expanding it with 1×1 and 3×3 filters for efficient feature extraction.

SqueezeNet primarily consists of eight sequential Fire modules, interspersed with max-pooling layers to reduce spatial dimensions. This architecture can serve as an efficient and effective fully convolutional backbone for feature extraction.

For the detection pipeline, inspired by YOLO [1], SqueezeDet employs a fully convolutional detector called ConvDet. After receiving the feature maps from the backbone, ConvDet generates bounding box coordinates and class probabilities across spatial locations. Subsequently, non-maximum suppression (NMS) eliminates overlapping predictions to yield the final detection results.

2.2. Feature-Based Video Object Detection

When using RNN-like networks to fuse information from video frames, two primary methods exist: box-level methods [27,30,31] and feature-based methods [28,29,32]. The key distinction lies in their input representations. Box-level methods typically apply RNNs to process bounding boxes generated by single-frame detectors, whereas feature-based methods utilize RNNs to operate on feature maps extracted from multiple frames. In this work, we adopt the feature-based approach to capture both temporal and spatial information.

Several feature-based methods have been proposed for video object detection [28,29,32]. Among these, due to its simplicity, effectiveness, and generalizable architecture, we select RMf-SSD [28] as our baseline model. The overall detection pipeline of RMf-SSD is illustrated in Figure 2.

RMf-SSD primarily consists of three components: a feature extractor, a recurrent unit serving as the data fusion layer, and a detection head. The key modification they introduced to the single-frame detection framework is the inclusion of a data fusion layer, which integrates temporal information from multiple frames. In their original research, various data fusion techniques were explored, including addition, max, and concatenation.

However, both empirical results and theoretical analysis demonstrate that using a recurrent unit as the fusion mechanism outperforms other methods.

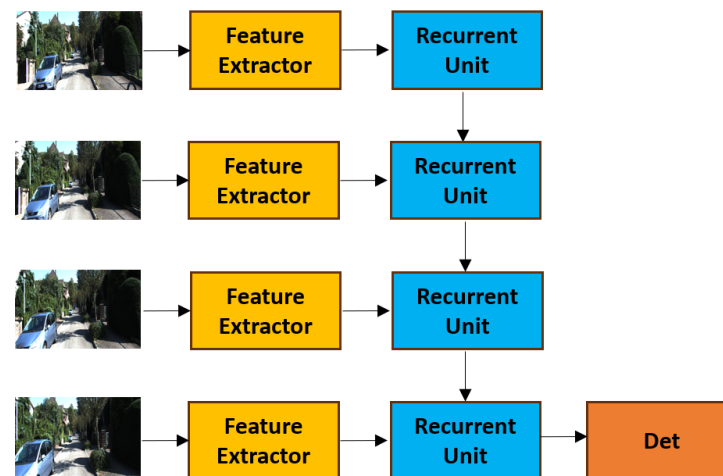


Figure 2. The architecture of RMf-SSD, which leverages a recurrent unit to integrate spatiotemporal features for enhanced video object detection.

The design of RMf-SSD is based on SqueezeDet, which in turn builds upon the SqueezeNet architecture. SqueezeNet is renowned for its highly efficient design, enabling RMf-SSD to achieve real-time detection despite the additional computational demands of processing video frames. In their experiments, alternative feature extractors such as VGG-16 and ResNet-50 were also tested. While these architectures demonstrated improved performance, their significantly larger number of parameters compared to SqueezeNet posed challenges in terms of memory requirements. To address these constraints, modifications were necessary, but these adjustments limited the extent of performance gains, making SqueezeNet a more practical choice overall.

For the feature fusion unit of the RMf-SSD framework, ConvGRU [36], a convolutional adaptation of the Gated Recurrent Unit (GRU), is utilized. ConvGRU primarily functions by substituting the dot-product operations inherent in conventional recurrent units with convolutional operations. Such a modification allows the model to exploit localized spatial information derived from convolutional processes, in contrast to the traditional GRU, which is limited to capturing global information. By employing ConvGRU to integrate features extracted from sequential frames, the approach generates a feature map that effectively encapsulates both spatial and temporal dimensions.

After integrating features across multiple frames via ConvGRU, the processed feature representations are subsequently input into the detection head to produce detection outputs. By synthesizing temporal context extracted from the video sequence with spatial details preserved from individual frames, RMf-SSD significantly outperforms its single-frame counterpart, achieving a substantial improvement in detection accuracy while maintaining real-time processing capabilities.

2.3. Neural ODE

Over the past few years, due to the way they process information, it has been recognized that traditional neural networks can be viewed as discrete approximations of Neural Ordinary Differential Equations (ODEs) [37]. This perspective has led to the development of various differential equation-based networks, which have demonstrated superior performance compared to their discretized counterparts in various tasks [38–40]. A seminal work in this field is Neural ODE [37], the formulation of which is defined as follows:

$$y(0) = y_0 \quad \frac{dy}{dt}(t) = f_{\theta}(y(t), t) \quad (1)$$

where y_0 is an arbitrary-dimensional tensor representing the initial condition for the time series data, and θ denotes the learned parameters of the neural network f_{θ} .

In Neural ODE, the hidden state $y(t)$ can be interpreted as the solution to an initial-value problem for an ordinary differential equation (ODE). This solution is obtained using an ODE solver:

$$y(t_1) = y(t_0) + \int_{t_0}^{t_1} f_{\theta}(y(t), t) dt = \text{ODESolve}(y(t_0), f_{\theta}, t_0, t_1). \quad (2)$$

From this equation, we can see that unlike traditional discrete-time models, which update the hidden state only at observation points, Neural ODEs define how the hidden state evolves continuously over time.

Neural ODEs are naturally suited for modeling continuous-time-based processes. However, a fundamental limitation of Neural ODEs lies in their dependence on initial conditions: the final solution $y(t_1)$ is entirely determined by the initial condition $y(t_0)$. This characteristic prevents the ODE solution from being adjusted using subsequent observations, making it unsuitable for online prediction. And this limitation is particularly problematic for real-time applications, such as our work, where immediate predictions based on incoming data are essential.

Several approaches have been proposed to address this limitation. For instance, Kidger et al. [39] introduced Neural Controlled Differential Equations (Neural CDEs), which use a control path X to guide the evaluation of $y(t_1)$. This allows the model to incorporate both initial and intermediate data points, enhancing its ability to handle continuous-time processes. Building on this, Morrill et al. [41] demonstrated that constructing specific control signals for Neural CDEs can enable them to perform online prediction tasks effectively. Furthermore, Morrill et al. [42] extended Neural CDEs by leveraging rough differential equations, enabling the model to handle long time-series data more robustly. Another notable approach is ODE-RNN, proposed by Rubanova et al. [40]. This model extends traditional RNNs by allowing their hidden states to evolve continuously between observations, making them more suitable for real-world time-based data. In this work, we aim to leverage similar ideas from ODE-RNN to enhance the overall performance of our detection model.

3. Proposed Method

The pipeline of the proposed NODE-ConvGRU hybrid model for video object detection is illustrated in Figure 3, which mainly includes following modules:

- (1) A novel pipeline for processing temporal features is introduced in the temporal features block to characterize the hidden states. In contrast to traditional recurrent neural networks (RNNs), which interpret input data as a discrete sequence, our approach utilizes a Neural ODE to model the hidden states between data observations; upon the arrival of new data, the ConvGRU updates the hidden state accordingly, thereby representing the hidden state as a continuous dynamical system throughout the overall framework.
- (2) An efficient and effective SqueezeNet-based feature extractor is employed to extract spatial features from input frames. Integrated with a feature fusion module, a bottom-up hierarchical structure fuses high-level semantic features with detailed spatial information to enhance the model's representational capacity.

- (3) A fully convolutional detector augmented with CBAM is designed to harness attention mechanisms, allowing the model to emphasize critical features in both channel and spatial domains, thereby enhancing feature representation and detection precision.

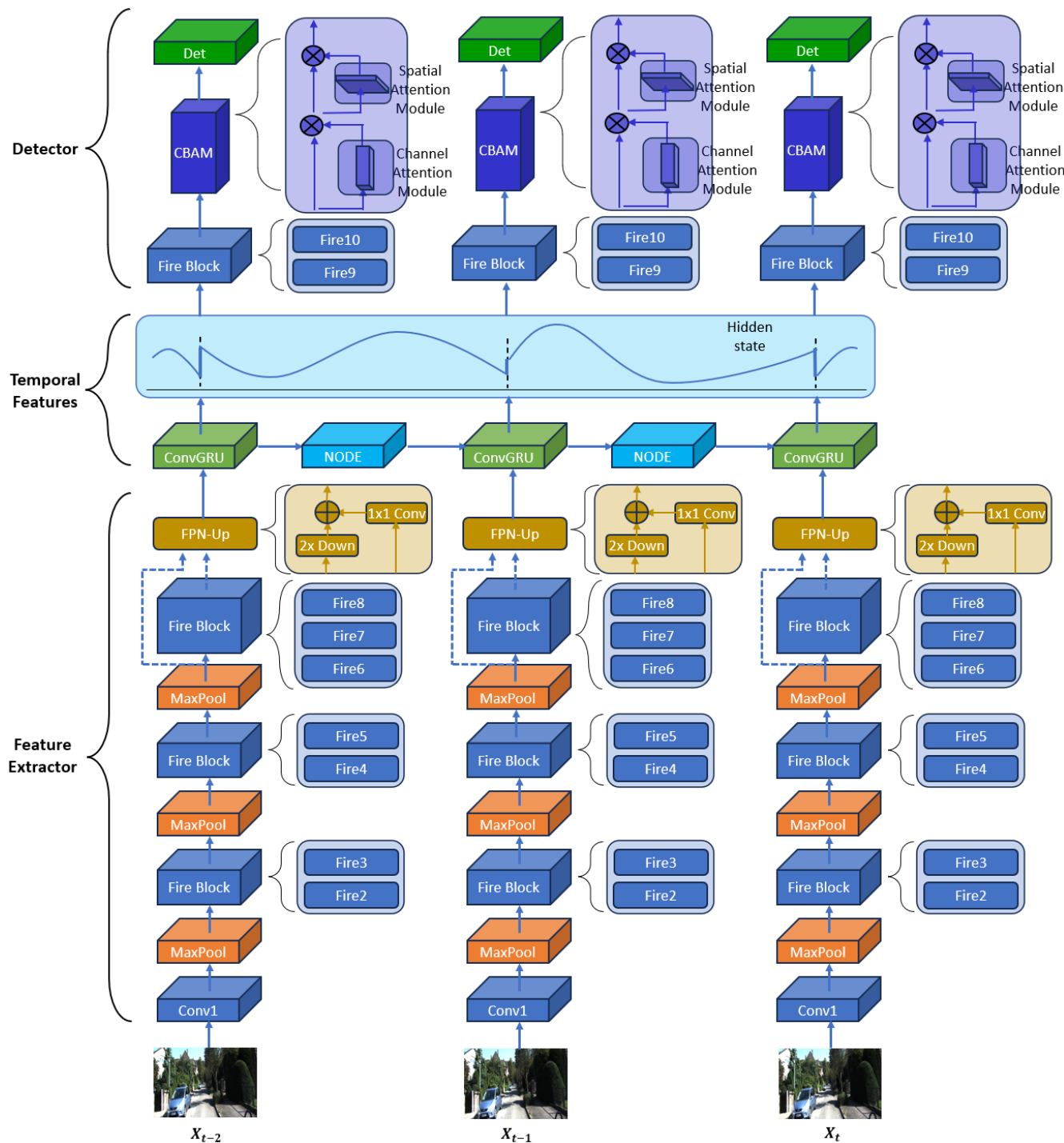


Figure 3. Overview of the hybrid detection pipeline. First, the input video frame is passed through a Feature Extractor to obtain feature maps, which serve as input to the ConvGRU. The ConvGRU then generates a hidden state h_{t_i} , which is used as the initial value by the ODE Solver, after which the solver computes intermediate hidden states h'_i between observations until the arrival of a new frame; then, ConvGRU integrates h'_i and the new frame's features, making it overall a continuous process. At each observation point, the model performs detection based on h_{t_i} .

3.1. NODE-ConvGRU Hybrid for Continuous Dynamics

Prior approaches [28,43,44] leverage temporal features from video data using a ConvGRU [36] architecture, defined as follows:

$$\begin{aligned} z_t &= \sigma(W_z * x_t + U_z * h_{t-1}), & r_t &= \sigma(W_r * x_t + U_r * h_{t-1}) \\ \tilde{h}_t &= \tanh(W * x_t + U * (r_t \odot h_{t-1})), & h_t &= (1 - z_t)h_{t-1} + z_t\tilde{h}_t \end{aligned} \quad (3)$$

where $*$ denotes the convolutional operator, and $\odot$ indicates the element-wise product. The terms z_t , r_t , $\tilde{h}_t$, h_t , and x_t represent the update gate, reset gate, candidate activation, and output and input feature maps, respectively.

While ConvGRU adeptly handles discrete frame inputs, it struggles to represent the continuous temporal dynamics of real-world video data and relies on an assumption of uniform time intervals, which does not always hold. To overcome these limitation, we introduce the NODE-ConvGRU architecture, integrating Neural ODEs with ConvGRU. This architecture enables continuous modeling of the network's hidden states, and it can naturally accommodate variable time intervals. The architecture can be formulated as follows:

$$\begin{aligned} h'_{t_i} &= \text{ODESolve}(f_\theta, h_{t_{i-1}}, (t_{i-1}, t_i)) \\ h_{t_i} &= \text{ConvGRU}(h'_{t_i}, E(X_{t_i})) \end{aligned} \quad (4)$$

where $X_{t_i} \in R^{w \times h \times c}$ represents a frame from the video data and $h_{t_i} \in R^{w' \times h' \times c'}$ represents the feature representation extracted at time t_i , with spatial dimensions $w' \times h'$ and c' channels.

The process proceeds as follows: the ODE Solver computes the hidden state h'_{t_i} before the next observation by integrating the neural network f_θ , which governs the evolution of the hidden state through the differential equation $f_\theta(t) = dh_{t_i}/dt$, with the initial hidden state set to zero by default. Given an input frame X_{t_i} , the feature extractor backbone E generates initial hidden states for the image, represented as feature tensor. Then this feature tensor, along with the hidden state h'_{t_i} , is passed to the ConvGRU, which computes the updated hidden state h_{t_i} as $h_{t_i} = \text{ConvGRU}(h'_{t_i}, E(X_{t_i}))$.

The architecture we choose can leverage the strengths of both ConvGRU and Neural ODEs. When observation data from a time-based process are received, the model updates the hidden state h_{t_i} using a ConvGRU. However, the input hidden state at this step is not identical to the previously obtained hidden state $h_{t_{i-1}}$ from ConvGRU. Instead, it utilizes h'_{t_i} , a hidden state derived and optimized through a Neural ODE network featuring continuous dynamics, so unlike fixed-state approaches, Neural ODEs can adaptively model the hidden state between observations, offering an efficient means to integrate continuous temporal dynamics into our framework. This design allows the model to naturally treat the entire process as a continuous online process, making it better suited for real-world time-based data.

Additionally, standard RNNs perform optimally when the time-series data have uniform intervals. However, the data we are working with may lack this property, especially in a real-world scenario. Our architecture can address this limitation by incorporating timestamps to explicitly capture the timing information, eliminating the need to preprocess or modify the data to fit the requirement. This approach preserves the integrity of the original data, avoiding potential information loss that could arise from such modifications [45].

3.2. Feature Pyramid Network-Up

To utilize a different scale of features from the backbone, we propose the use of an FPN-Up module to fuse multi-scale features from different layers of the backbone. Such

an approach enables our model to better leverage the distinct contributions of features extracted at various levels of the backbone network. The meta architecture of FPN-Up is illustrated in Figure 4.

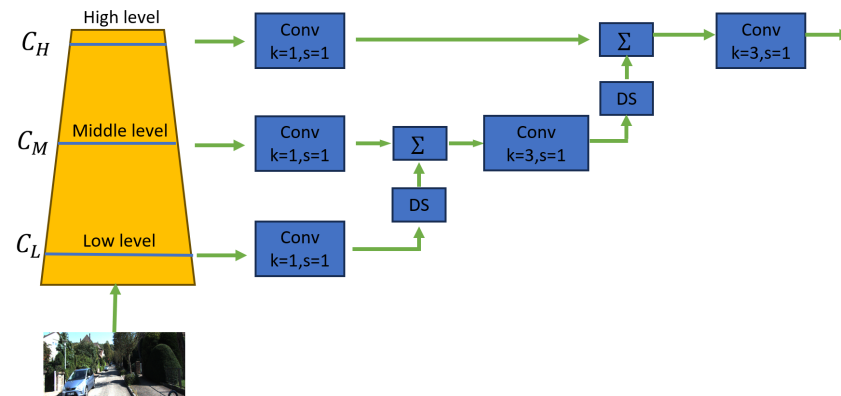


Figure 4. The network structure of FPN-Up.

The overall process can be defined as follows:

$$\begin{aligned} P_i &= \text{Conv}_{1 \times 1}^D(C_i) \\ P_{i+1} &= \text{Conv}_{3 \times 3}^D(P_{i+1} + \text{Downsample}(P_i)) \end{aligned} \quad (5)$$

where $i = L, M, H$.

Drawing inspiration from traditional FPNs, our approach leverages lateral connections to facilitate feature fusion across different scales. Specifically, we first apply a 1×1 convolution to transform the feature layers to a uniform target dimension D , yielding transient features P_i . Next, higher-resolution feature maps are downsampled via max pooling to achieve spatial alignment with deeper-layer features. Subsequently, an element-wise addition is performed to integrate these features, followed by a convolutional operation to refine and fuse the resulting representations. We use the fused feature map as the input to the next layer.

In conventional backbone architectures, deeper layers generally extract high-level, abstract semantic information, while shallower layers preserve detailed spatial features, often associated with smaller-scale objects. However, in the original RMf-SSD implementation, object detection relies exclusively on the final feature layer. This design choice introduces notable limitations, particularly in scenarios involving small-sized targets such as pedestrians or distant objects, where precise localization is paramount. However, it is important to note that SqueezeNet, the backbone architecture employed in this context, prioritizes parameter efficiency and exhibits a comparatively shallow structure relative to architectures such as ResNet-50 or VGG-16. As a result, integrating a full Feature Pyramid Network (FPN) [46] or BiFPN [47] for multi-scale feature extraction may not yield commensurate benefits for this specific backbone. Our simplified process emphasizes the enrichment of high-level semantic features with additional spatial context, enhancing the model's capacity for precise object localization and improving prediction accuracy while still preserving model efficiency.

3.3. Convolutional Block Attention Module (CBAM)

Attention mechanisms form a fundamental component of contemporary deep learning, allowing models to prioritize the most salient aspects of input data and thus enhance overall performance. In this work, we integrate the CBAM module [33] into our detector to leverage its attention capabilities. Introduced by Woo et al., CBAM is a compact and efficient plug-in

module delivering a robust attention mechanism. It improves CNN feature representation by enabling the model to emphasize key regions across channel and spatial dimensions. The architecture of CBAM is illustrated in Figure 5. The working process of CBAM is as follows: Given an input feature map F , the module first generates a channel-wise attention map M_c . This map is then applied to F to produce a channel-refined feature map F' . Subsequently, a spatial-wise attention map M_s is generated from F' and then been applied to F' to refine F' further, resulting in the final refined feature map F'' . This dual-attention approach enables the model to enhance features across both channels and spatial regions within the feature maps. The process can be mathematically defined as follows:

$$\begin{aligned} F' &= M_c(F) \otimes F \\ F'' &= M_s(F') \otimes F' \end{aligned} \quad (6)$$

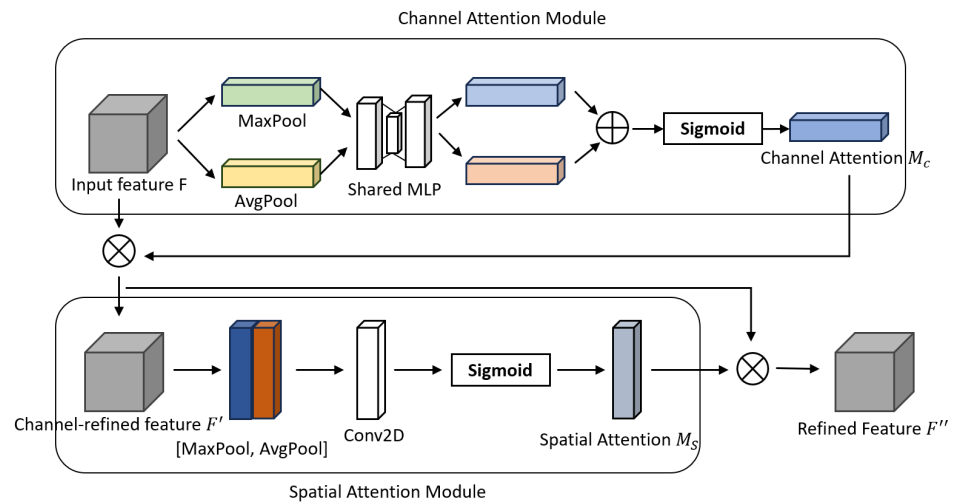


Figure 5. Convolutional block attention module.

In the CBAM architecture, the channel attention module [33] emphasizes the most informative channels by learning to assign weights based on their relevance to the task. First, it will apply global average pooling and global max pooling across the spatial dimensions of the input to generate channel-wise descriptors. These descriptors are processed by a shared multi-layer perceptron (MLP) to generate outputs from two branches, which are subsequently summed and passed through a sigmoid activation function to yield the final channel attention weights. These weights are then used to scale the original feature maps, enhancing the important channels while suppressing less relevant ones. The process can be mathematically described as follows:

$$M_c(F) = \sigma(MLP(AvgPool(F)) + MLP(MaxPool(F))) \quad (7)$$

On the other hand, the spatial attention module [33] focuses on identifying the most relevant spatial locations within the feature maps. It operates by first generating a spatial-wise feature descriptor through average pooling and max pooling across the channel dimension. The pooled features are combined into a feature descriptor, which is then processed by a 7×7 convolutional layer followed by a sigmoid activation function to generate the final spatial attention map. This map is subsequently used to enhance feature maps by highlighting key spatial areas. The process can be mathematically described as follows:

$$M_s(F) = \sigma(f^{7 \times 7}([AvgPool(F); MaxPool(F)])) \quad (8)$$

4. Experiment Results

4.1. Experimental Setup and Dataset Preparation

The experiments were conducted on an NVIDIA GeForce RTX 4000 GPU (20 GB VRAM) using Ubuntu 22.04, PyTorch 2.4.1, and Torchvision 0.19.1, with CUDA 11.8 for GPU acceleration. ODEs were solved using torchdiffeq (v0.2.4) within PyTorch.

We evaluate the performance of our proposed model on the KITTI object detection dataset [48], a widely used benchmark for video object detection in the context of autonomous driving and computer vision. The dataset consists of roughly six hours of driving data gathered from urban, rural, and highway settings. Most images feature multiple objects across various classes, often including numerous small objects exhibiting diverse levels of occlusion, saturation, shadows, and truncation. This composition offers a faithful depiction of real-world conditions. The dataset is annotated for three classes: cars, pedestrians, and cyclists.

In the KITTI object detection dataset, objects are divided into three difficulty levels—Easy, Moderate, and Hard—based on criteria such as size, occlusion, and truncation, with harder difficulty levels encompassing objects from easier ones due to less restrictive criteria; for instance, “Easy” objects have a minimum height of 40 pixels, whereas “Moderate” reduces this threshold to 25 pixels. The 2D object detection subset includes 7481 training images and 7518 testing images, all captured with a resolution of 1242×375 . In the static image object detection task, each image in the KITTI 2D object detection dataset is annotated with bounding boxes and class labels for detection targets. For video object detection, the dataset provides three preceding frames for each detection frame, all captured at a rate of 10 Hz. Although these preceding frames are not labeled, their features can be extracted using the same feature extractor, making them suitable for feature-based video object detection tasks. For approximately 30 detection frames in the dataset where not all three preceding frames are available, we pad the missing frames with the nearest available frames in time to maintain consistency during training. For this experiment, we randomly split the training data into two equal parts to create training and validation sets. All models are trained on the same train/val split to ensure a fair comparison.

4.2. Experimental Parameters and Evaluation Metrics

Concerning the training parameters, the input images were resized to a resolution of 1248×384 . To prevent overfitting, we employed several data augmentation techniques, such as scaling, rotation, image normalization. We initialized the backbone using the official SqueezeNet pre-trained weights on the ImageNet dataset. For the training process, first, we constructed an RMf-SSD model and trained it for 300 epochs on the KITTI dataset. Subsequently, we utilized the trained weights obtained from RMf-SSD as initialization for our proposed detection pipeline to explore its potential. We used Stochastic Gradient Descent with momentum to optimize the loss function. We set the epochs to 300, initial learning rate to 0.01, learning rate decay factor to 0.5, and decay step to 60. We utilized the KITTI development kit to evaluate the performance of our models.

To evaluate the performance of our model, we employ precision, recall, and mean Average Precision (mAP) as key metrics, while Frames Per Second (FPS) is utilized to assess the model’s real-time processing capabilities. Precision and recall are defined as follows:

$$Precision = \frac{TP}{TP + FP} \quad (9)$$

$$Recall = \frac{TP}{TP + FN} \quad (10)$$

where True Positives (TP) represents the number of targets correctly detected, False Positives (FP) denotes the number of incorrect detections, and False Negatives (FN) indicates the number of labeled targets missed by the detector. Precision quantifies the proportion of true detections among all detections made by the model, reflecting its accuracy, while recall measures the proportion of ground-truth targets successfully identified, indicating the model's completeness. By plotting precision against recall, we can construct the precision–recall (PR) curve, a critical tool for calculating the Average Precision (AP). As for the mAP , it is computed by first determining the AP for each class, which involves integrating the PR curve, and then averaging these values across all classes. This can be mathematically expressed as follows:

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i = \frac{1}{N} \sum_{i=1}^N \int_0^1 P(r) dr \quad (11)$$

4.3. Experiment Result

To validate the performance of the proposed model, we conducted experiments on both the RMf-SSD and proposed model. The detection accuracy, measured by mAP , is presented in Table 1. Our proposed model achieves the highest mAP across all classes.

Table 1. Detection performance. Class-specific mAP is computed as an average over varying difficulty levels. The overall mAP reflects an average across all classes and difficulty levels.

Method	Car mAP	Cyclist mAP	Pedestrian mAP	All mAP	FPS
RMf-SSD	88.1	81.0	79.4	82.9	65
Proposed Model	91.6	84.8	80.6	85.7	58
SqueezeDet	84.2	77.8	75.8	79.3	71

As illustrated in Table 1, the proposed NODE-ConvGRU hybrid model achieves a 2.8% improvement in detection accuracy across all classes and difficulty levels compared to the RMf-SSD model. For benchmarking purposes, we also evaluated the performance of a vanilla SqueezeDet model. The results presented in Table 1 demonstrate that incorporating multi-frame feature extraction to capture temporal dependencies significantly enhances model performance. However, our proposed model further refines this approach by more effectively leveraging temporal and spatial information, resulting in superior accuracy. A comprehensive breakdown of the accuracy metrics is provided in Table 2.

Table 2. Detailed detection results.

Method	Car			Cyclist			Pedestrian			All mAP
	E	M	H	E	M	H	E	M	H	
RMf-SSD	94.6	89.4	80.3	85.8	78.9	78.4	85.6	77.5	75.1	82.9
Proposed Model	96.5	90.1	88.2	87.6	87.3	79.6	87.1	78.3	76.6	85.7
SqueezeDet	89.5	86.5	76.7	81.6	77.3	74.5	82.1	74.2	71.2	79.3

In terms of real-time performance, our proposed hybrid model delivers detection at 58 frames per second (FPS) during inference, indicating comparable speeds across all three models. This similarity in inference speed can be primarily attributed to the architectural design of the proposed model, which processes video frames sequentially during the inference stage. Consequently, video frames can be processed by the network in real time as they are acquired. Although our model incorporates additional computational components to better leverage real-world time-series data, these enhancements do not compromise its ability to operate in real time. Instead, they contribute to a significant improvement in performance while maintaining real-time capabilities.

Figure 6 presents the PR curve to illustrate our model's performance. The KITTI development kit streamlines AP calculations by setting precision to 0 beyond the maximum achievable recall. The PR curves showcase our model's consistent superiority over the baseline RMf-SSD across all object classes and difficulty levels. For instance, consider the Moderate difficulty category for cyclists as a specific example. Here, our model achieves a higher maximum recall, which means it is capable of detecting a greater number of objects compared to the baseline. Additionally, at equivalent recall levels, our model sustains higher precision, indicating that it delivers more accurate detections by correctly identifying true positives while reducing false positives. This combination of enhanced recall and precision underscores the overall effectiveness of our proposed model in object detection tasks.

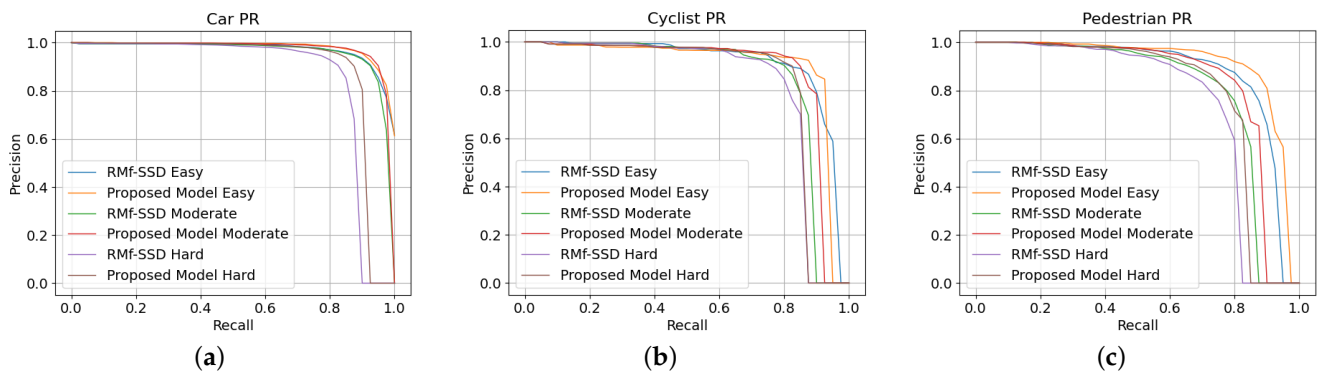


Figure 6. Comparison of PR curve for three different classes, each comprised of Easy, Moderate, and Hard difficulty levels. (a) Detection comparison for Car; (b) detection comparison for Cyclist; (c) detection comparison for Pedestrian.

To provide a more intuitive comparison between the proposed model and RMf-SSD, Figure 7 presents detection results from some complex real-world scenarios. Upon analyzing the visualization results, we can conclude that our proposed model exhibits superior performance in detecting pedestrians at greater distances and in densely clustered regions with overlapping objects. These findings underscore the effectiveness of our approach in addressing challenging detection tasks.

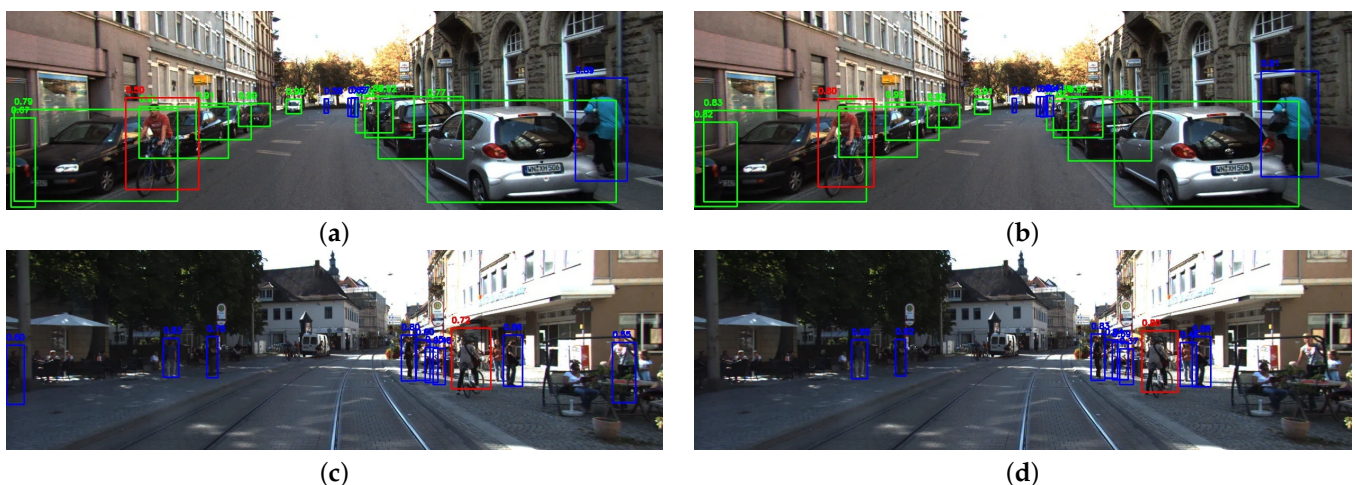


Figure 7. Some detection results of RMf-SSD and proposed model. It is evident that our proposed model's detection performance is much better than RMf-SSD. (a) Detection results of RMf-SSD. (b) Proposed model detection results. (c) Detection results of RMf-SSD model. (d) Proposed model detection results.

4.4. Ablation Study

We conducted an ablation study on our proposed model to evaluate the contribution of each modification to the overall performance improvement. We use Floating-Point Operations per Second (FLOPs) and number of parameters as metrics to assess the computational cost of each model. FLOPs mainly measures the computational complexity of the model, while the number of parameters indicates the model's memory footprint. The results of this analysis are presented in Table 3.

Table 3. Detection accuracy. The mAP for different classes is averaged across different difficulty levels. The mAP for All is averaged across all classes and difficulty levels.

Method	NODE	FPN-Up	CBAM	Car mAP	Cyclist mAP	Pedestrian mAP	All mAP	GFLOPs	Param.
RMf-SSD				88.1	81.0	79.4	82.9	61.92	14.15 M
RMf-SSD + NODE	✓ ¹			90.9	83.1	79.8	84.6	72.96	17.09 M
RMf-SSD + NODE + FPN-Up	✓	✓		91.3	84.0	80.5	85.3	83.26	19.84 M
RMf-SSD + NODE + CBAM	✓		✓	91.2	83.9	80.3	85.1	72.97	17.09 M
Proposed Model	✓	✓	✓	91.6	84.8	80.6	85.7	83.27	19.84 M

¹ ✓ indicates the module used.

From Table 3, it is evident that incorporating a Neural ODE into our model to enable continuous dynamics can significantly improve detection performance, with a computational cost of 72.96 GFLOPs and an additional 2.94 M parameters. Specifically, our proposed architecture enhances detection performance by 1.7 mAP compared to RMf-SSD, and while RMf-SSD achieves a 3.6 mAP improvement over SqueezeDet, our model can deliver a total gain of 5.3 mAP relative to the original SqueezeDet framework. This result underscores the importance of reflecting continuous dynamics in the model, as it leads to a notable enhancement in detection accuracy. Subsequently, the addition of our feature fusion layer further boosts performance by 0.7 mAP with a computational cost of 83.26 GFLOPs and an additional 2.75 M parameters. This improvement is particularly pronounced for detecting pedestrians and cyclists, highlighting the benefits of integrating high-level semantic features with low-level spatial information. Furthermore, the incorporation of the CBAM enables the model to focus on the most relevant regions of the input data, yielding an additional performance gain of 0.5 mAP with a computational cost of 72.97 GFLOPs and 17.09 M parameters. As demonstrated in the results, this attention mechanism provides substantial improvements, underscoring its effectiveness in enhancing feature representation and detection accuracy. Finally, the comprehensive utilization of all the components proposed in our model yields an overall performance improvement of 2.8%, with a total computational cost of 83.27 GFLOPs and 19.84M parameters, while still performing in real time, demonstrating the efficacy and utility of our proposed approach.

5. Conclusions

In this study, we introduced a novel NODE-ConvGRU hybrid model that can leverage continuous hidden states for video object detection tasks. Unlike previous methods that rely on discrete updates, our model's use of continuous dynamics enables a more natural alignment with real-world time-based processes. Additionally, we incorporated a feature fusion layer to effectively combine high-level semantic features with low-level spatial details, thereby enhancing the utilization of multi-layer features. Moreover, a CBAM was embedded within the detection head to prioritize the most salient regions of the feature maps. Experimental results indicate that these methodological innovations substantially improve performance in video object detection tasks while still performing in real time. While utilizing a standard ODE formulation to capture the continuous dynamics of video data can significantly improve the detection results, considering the substantial body of

ODE research, we plan to explore alternative ODE-driven designs in future work, possibly more suited to video data, to further elevate performance. Additionally, in our experiment, for computational resource and temporal constraint reasons, we only assessed the efficacy of our proposed model using the KITTI object detection dataset; in the future, we plan to further evaluate the robustness of our model by applying it to additional datasets and implementing it in a real-world scenario for intelligent inspection purposes.

Author Contributions: Conceptualization, L.Q.; data curation, L.Q.; formal analysis, L.Q.; funding acquisition, S.S. and S.L.; investigation, L.Q.; methodology, L.Q.; project administration, S.S. and S.L.; resources, L.Q.; software, L.Q.; supervision, S.S. and S.L.; validation, L.Q.; visualization, L.Q.; writing—original draft, L.Q.; writing—review and editing, L.Q., S.S. and S.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Key Research and Development Program of Guangxi under grant number AB24010110.

Data Availability Statement: Relevant data are contained within the article.

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

References

1. 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.
2. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In Proceedings of the Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; Proceedings, Part I 14; Springer: Berlin/Heidelberg, Germany, 2016; pp. 21–37.
3. 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.* **2016**, *39*, 1137–1149. [[CrossRef](#)] [[PubMed](#)]
4. Wu, B.; Iandola, F.; Jin, P.H.; Keutzer, K. Squeezednet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Honolulu, HI, USA, 21–26 July 2017; pp. 129–137.
5. Zou, Z.; Chen, K.; Shi, Z.; Guo, Y.; Ye, J. Object detection in 20 years: A survey. *Proc. IEEE* **2023**, *111*, 257–276. [[CrossRef](#)]
6. Kaur, R.; Singh, S. A comprehensive review of object detection with deep learning. *Digit. Signal Process.* **2023**, *132*, 103812. [[CrossRef](#)]
7. Jiao, L.; Zhang, R.; Liu, F.; Yang, S.; Hou, B.; Li, L.; Tang, X. New generation deep learning for video object detection: A survey. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *33*, 3195–3215. [[CrossRef](#)]
8. Yang, X.; Yuanfang, Q.; Delin, L.; Haibin, D.; Zhengyu, G. Distributed predefined-time estimator-based affine formation target-enclosing maneuver control for cooperative underactuated quadrotor UAVs with fault-tolerant capabilities. *Chin. J. Aeronaut.* **2025**, *38*, 103042.
9. Luo, D.; Shao, J.; Xu, Y.; Zhang, J. Docking navigation method for UAV autonomous aerial refueling. *Sci. China Inf. Sci.* **2019**, *62*, 1–16. [[CrossRef](#)]
10. Yuan, Z.; Song, X.; Bai, L.; Wang, Z.; Ouyang, W. Temporal-channel transformer for 3d lidar-based video object detection for autonomous driving. *IEEE Trans. Circuits Syst. Video Technol.* **2021**, *32*, 2068–2078. [[CrossRef](#)]
11. Zhu, H.; Wei, H.; Li, B.; Yuan, X.; Kehtarnavaz, N. A review of video object detection: Datasets, metrics and methods. *Appl. Sci.* **2020**, *10*, 7834. [[CrossRef](#)]
12. Xie, X.; Cheng, G.; Wang, J.; Yao, X.; Han, J. Oriented R-CNN for object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 3520–3529.
13. Liu, L.; Ouyang, W.; Wang, X.; Fieguth, P.; Chen, J.; Liu, X.; Pietikäinen, M. Deep learning for generic object detection: A survey. *Int. J. Comput. Vis.* **2020**, *128*, 261–318. [[CrossRef](#)]
14. Hmidani, O.; Alaoui, E.I. A comprehensive survey of the R-CNN family for object detection. In Proceedings of the 2022 5th International Conference on Advanced Communication Technologies and Networking (CommNet), Marrakech, Morocco, 12–14 December 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–6.
15. Sun, P.; Zhang, R.; Jiang, Y.; Kong, T.; Xu, C.; Zhan, W.; Tomizuka, M.; Li, L.; Yuan, Z.; Wang, C.; et al. Sparse r-cnn: End-to-end object detection with learnable proposals. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 14454–14463.

16. Hussain, M. YOLO-v1 to YOLO-v8, the rise of YOLO and its complementary nature toward digital manufacturing and industrial defect detection. *Machines* **2023**, *11*, 677. [\[CrossRef\]](#)
17. Diwan, T.; Anirudh, G.; Tembhurne, J.V. Object detection using YOLO: Challenges, architectural successors, datasets and applications. *Multimed. Tools Appl.* **2023**, *82*, 9243–9275. [\[CrossRef\]](#)
18. Li, M.; Li, L.; Bai, R.; Ren, J.; Meng, B.; Yang, Y. A motion-based seq-bbox matching method for video object detection. In Proceedings of the 2021 IEEE Symposium on Computers and Communications (ISCC), Athens, Greece, 5–8 September 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–7.
19. Lopez-Marcano, S.; L Jinks, E.; Buelow, C.A.; Brown, C.J.; Wang, D.; Kusy, B.; M Ditria, E.; Connolly, R.M. Automatic detection of fish and tracking of movement for ecology. *Ecol. Evol.* **2021**, *11*, 8254–8263. [\[CrossRef\]](#)
20. Wu, J.; Cao, J.; Song, L.; Wang, Y.; Yang, M.; Yuan, J. Track to detect and segment: An online multi-object tracker. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 12352–12361.
21. Zhang, M.; Liu, J.; Wang, Y.; Piao, Y.; Yao, S.; Ji, W.; Li, J.; Lu, H.; Luo, Z. Dynamic context-sensitive filtering network for video salient object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 1553–1563.
22. Fan, L.; Zhang, T.; Du, W. Optical-flow-based framework to boost video object detection performance with object enhancement. *Expert Syst. Appl.* **2021**, *170*, 114544. [\[CrossRef\]](#)
23. Zhou, Q.; Li, X.; He, L.; Yang, Y.; Cheng, G.; Tong, Y.; Ma, L.; Tao, D. TransVOD: End-to-end video object detection with spatial-temporal transformers. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *45*, 7853–7869. [\[CrossRef\]](#) [\[PubMed\]](#)
24. He, L.; Zhou, Q.; Li, X.; Niu, L.; Cheng, G.; Li, X.; Liu, W.; Tong, Y.; Ma, L.; Zhang, L. End-to-end video object detection with spatial-temporal transformers. In Proceedings of the 29th ACM International Conference on Multimedia, Virtual, 20–24 October 2021; pp. 1507–1516.
25. Cui, Y.; Yan, L.; Cao, Z.; Liu, D. Tf-blender: Temporal feature blender for video object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 8138–8147.
26. Xu, C.; Zhang, J.; Wang, M.; Tian, G.; Liu, Y. Multilevel spatial-temporal feature aggregation for video object detection. *IEEE Trans. Circuits Syst. Video Technol.* **2022**, *32*, 7809–7820. [\[CrossRef\]](#)
27. Chen, K.; Wang, J.; Yang, S.; Zhang, X.; Xiong, Y.; Loy, C.C.; Lin, D. Optimizing video object detection via a scale-time lattice. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7814–7823.
28. Broad, A.; Jones, M.; Lee, T.Y. Recurrent Multi-frame Single Shot Detector for Video Object Detection. In Proceedings of the British Machine Vision Conference, Newcastle, UK, 3–6 September 2018; p. 94.
29. Liu, M.; Zhu, M. Mobile video object detection with temporally-aware feature maps. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 5686–5695.
30. Tripathi, S.; Lipton, Z.C.; Belongie, S.; Nguyen, T. Context matters: Refining object detection in video with recurrent neural networks. *arXiv* **2016**, arXiv:1607.04648.
31. Ning, G.; Zhang, Z.; Huang, C.; Ren, X.; Wang, H.; Cai, C.; He, Z. Spatially supervised recurrent convolutional neural networks for visual object tracking. In Proceedings of the 2017 IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, USA, 28–31 May 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–4.
32. Yi, K.; Jian, Z.; Chen, S.; Zheng, N. Feature selective small object detection via knowledge-based recurrent attentive neural network. *arXiv* **2018**, arXiv:1803.05263.
33. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. Cbam: Convolutional block attention module. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.
34. Zhuang, J.; Chen, W.; Huang, X.; Yan, Y. Band Selection Algorithm Based on Multi-Feature and Affinity Propagation Clustering. *Remote Sens.* **2025**, *17*, 193. [\[CrossRef\]](#)
35. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size. *arXiv* **2016**, arXiv:1602.07360.
36. Ballas, N.; Yao, L.; Pal, C.; Courville, A. Delving deeper into convolutional networks for learning video representations. *arXiv* **2015**, arXiv:1511.06432.
37. Chen, R.T.; Rubanova, Y.; Bettencourt, J.; Duvenaud, D.K. Neural ordinary differential equations. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 1–13.
38. Lechner, M.; Hasani, R.; Amini, A.; Henzinger, T.A.; Rus, D.; Grosu, R. Neural circuit policies enabling auditable autonomy. *Nat. Mach. Intell.* **2020**, *2*, 642–652. [\[CrossRef\]](#)
39. Kidger, P.; Morrill, J.; Foster, J.; Lyons, T. Neural controlled differential equations for irregular time series. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 6696–6707.
40. Rubanova, Y.; Chen, R.T.; Duvenaud, D.K. Latent ordinary differential equations for irregularly-sampled time series. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 1–11.

41. Morrill, J.; Kidger, P.; Yang, L.; Lyons, T. Neural controlled differential equations for online prediction tasks. *arXiv* **2021**, arXiv:2106.11028.
42. Morrill, J.; Salvi, C.; Kidger, P.; Foster, J. Neural rough differential equations for long time series. In Proceedings of the International Conference on Machine Learning, Virtual, 18–24 July 2021; PMLR: Breckenridge, CO, USA, 2021; pp. 7829–7838.
43. Qasim Gandapur, M.; Verdú, E. ConvGRU-CNN: Spatiotemporal deep learning for real-world anomaly detection in video surveillance system. *Int. J. Interact. Multimed. Artif. Intell.* **2023**, *8*, 88–95. [[CrossRef](#)]
44. Yang, G.; Zhang, Y.; Li, W.; Tang, Y.; Shang, W.; Wen, F.; Zhang, H.; Ding, M. Geo-ConvGRU: Geographically Masked Convolutional Gated Recurrent Unit for Bird-Eye View Segmentation. *arXiv* **2024**, arXiv:2412.20171.
45. Weerakody, P.B.; Wong, K.W.; Wang, G.; Ela, W. A review of irregular time series data handling with gated recurrent neural networks. *Neurocomputing* **2021**, *441*, 161–178. [[CrossRef](#)]
46. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.
47. Tan, M.; Pang, R.; Le, Q.V. Efficientdet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 10781–10790.
48. Geiger, A.; Lenz, P.; Urtasun, R. Are we ready for autonomous driving? The kitti vision benchmark suite. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 3354–3361.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.