Object Detection for Autonomous Driving: Motion-aid Feature Calibration Network

Dongfang Liu, Yaqin Wang and Eric T. Matson
Computer and Information Technology, Purdue University, West Lafayette, IN, U.S.A.

Keywords: Autonomous Driving, Video Object Detection, Pixel Feature Calibration, Instance Feature Calibration.

Abstract: Object detection is a critical task for autonomous driving. The latest progress in deep learning research for object detection has built a solid contribution to the development of autonomous driving. However, direct employment of the state-of-the-art object detectors from image to video is problematic. Object appearances in videos have more variations, e.g., video defocus, motion blur, truncation, etc. Such variations of objects in a video have fewer occurrences in still image and could compromise the detection results. To address these problems, we build a fast and accurate deep learning framework, motion-assist calibration network (MFCN) for video detection. Our model leverages the motion pattern of temporal coherence on video features. It calibrates and aggregates features of detected objects from previous frames along the spatial changes to improve the feature representations on the current frame. The whole model architecture is trained end-to-end which boosts the detection accuracy. Validations on the KITTI and ImageNet dataset show that MFCN can improve the baseline results by 10.3% and 9.9% mAP respectively. Compared with other state-of-the-art models, MFCN achieves a leading performance on KITTI benchmark competition. Results indicate the effectiveness of the proposed model which could facilitate the autonomous driving system.

1 INTRODUCTION

Object detection is the primary task for the perception of autonomous car in order to achieve full autonomy(Liu et al., 2019)(Wang et al., 2019)(Ramanagopal et al., 2018). In recent years, the development of deep learning network has rendered a great progress in object detection which can be adapted by autonomous driving(Ren et al., 2015)(Lin et al., 2017b)(Zeng et al., 2016). The current state-of-the-art detectors are able to achieve recognizable success on static images (Chen et al., 2017), but their performance on live-stream videos needs to be improved greatly in order to feed the needs of autonomous driving(Zhu et al., 2017b). For autonomous driving, camera sensor constantly produces a live-stream video frame which may include a large number of object appearances with drastic variations. For instance, motion blur or video defocus are frequently observed. Under these circumstances, still Object detection may fail and generate unstable results.

To address these challenges in video object detection, one of intuitive solutions is to analyze the temporal and spatial coherence in videos and employ information from nearby frames(Du et al., 2017). Since videos generally include rich imagery information, it is easy to identify a same object instance in multiple frames in a short time. (Kang et al., 2018)(Han et al., 2016)(Lee et al., 2016) exploit such temporal information in existing video to detect objects in a simple way. These studies firstly utilize object detectors in individual frame and then assemble the inference results in a dedicated post processing step across all the temporal dimensions. Following the same principle, (Han et al., 2016)(Kang et al., 2018)(Kang et al., 2016)(Feichtenhofer et al., 2017) propose handcrafted bounding box association rules to improve the final predictions. Although the approaches mentioned above generate promising results, these methods are post-processing fashions not principled learning techniques.

To address the limitations of existing work, we further exploit the temporal information and seek more robust feature calibration to boost the accuracy of video object detection. In this study, we present motion-assist calibration network (MFCN), a fast, accurate, end-to-end framework for video object detection. We consider the effects of movement on features to facilitate the per-frame feature learning. We leverage the temporal coherence and calibrate pixel

Liu, D., Wang, Y. and Matson, E.
Object Detection for Autonomous Driving: Motion-aid Feature Calibration Network.
DOI: 10.5220/0009107602320239
In *Proceedings of the 12th International Conference on Agents and Artificial Intelligence (ICAART 2020) - Volume 2*, pages 232-239
ISBN: 978-989-758-395-7; ISSN: 2184-433X

and instance features across frames to prompt feature representation of current frame. Accordingly, the final detection is effectively improved.

The key contributions of our work are in threefold: First, we introduce a fast, accurate, end-to-end framework for video object detection which could be adopted by autonomous driving system; Second, we use ablation study to verify the improvements of the proposed model from a strong single-frame baseline and contributions of each module in the proposed model; and third, we conduct extensive experiments to evaluate the performance of the proposed model by comparing with state-of-the-art systems. Results indicate our model is competitive with the exiting best approaches for video object detection.

The rest of the paper is structured as follows. In Section II, a through discussion of related work is presented which builds the foundations of this research. Next, we articulate the technical details of the proposed model in Section III. In Section IV, we present the results of extensive experiments of our proposed model on two credential datasets. Finally, in section V, we conclude this study and address future work.

2 RELEVANT RESEARCH

With the development of deep learning for computer vision, the object detection has been extended from still image to video(Zhu et al., 2017b)(Zhu et al., 2018)(Zhang et al., 2018). One example is that the video object detection competition, such as ImageNet(Russakovsky et al., 2015), has drawn a wide attention in both academia and industry. For video object detection, there are two typical approaches from exiting work: 1. post processing; and 2. principed learning. We elaborate them in the following paragraph.

2.1 Post Processing

Many existing work leveraged state-of-the-art single image object detection systems to tackle the issue of object detection in video domain. (Uijlings et al., 2013)(Zitnick and Dollár, 2014) directly applied object detectors on individual frame and then assembled the inference results to generate the final predictions. These explorations were effective but naiive because they completely ignored the temporal dimension from the sequential video frames. On the contrary, (Han et al., 2016)(Lee et al., 2016)(Feichtenhofer et al., 2017) incorporated temporal information during the post-processing and effectively improved the detection results within each individual frame. Once region proposals and their corresponding class scores for each frame were obtained, (Han et al., 2016)(Lee et al., 2016)(Feichtenhofer et al., 2017) implemented algorithm to select boxes to maximize a sequence score. The selected boxes are then employed to suppress overlapping boxes in the corresponding frames in order to boost weaker detection. The post processing approach provided valuable insights for applications related to video surveillance. However, due to the nature of its off line design, post processing approach casts a cloud over the object detection for autonomous driving because this approach cannot be adopted directly for real-time driving system. In order to facilitate the autonomous driving, we seek answers from principled learning approach which can have real-time inferences for object detection on videos.

2.2 Principled Learning

Current state-of-the-art detectors generally embrace an identical two-stage structure. First, Deep Convolutional Neural Networks (CNNs) capture the outstanding features of the input image and generate a set of intermediate convolutional feature maps accordingly(Krizhevsky et al., 2012)(Simonyan and Zisserman, 2014)(He et al., 2016). Then, a detection network predicts the detection results based on the feature maps (Lin et al., 2017a)(He et al., 2015)(Liu et al., 2016). The first stage counts for the major computation cost compared to the final detection task. The intermediate convolutional feature maps have the same spatial extents of the input image but much smaller resolution. Such small resolution feature maps could be cheaply propagated by spatial warping from nearby frames (Ren et al., 2017) (Ma et al., 2015). To leverage this property of CNN, (Zhu et al., 2017b)(Kang et al., 2016)(Kang et al., 2018)(Zhu et al., 2017a) proposed end-to-end framework which used the feature propagation to enhance the feature of individual frames in videos. (Zhu et al., 2017b)(Zhu et al., 2017a) applied optical flow network (Ilg et al., 2017) to estimate per-pixel motion. Since the optical flow estimation and feature propagation had much faster computation than that of convolutional network feature extractions, (Zhu et al., 2017b)(Zhu et al., 2017a) avoided the computational bottleneck and achieve significant runtime improvement. Since both the motion flow and object detection were predicted by deep learning networks, the entire architecture was trained end-to-end and the detection accuracy and efficiency were significantly boosted (Zhu et al., 2017b)(Zhu et al., 2017a). Similarly, (Kang et al., 2016) (Kang et al., 2018) presented a novel work using tubelet proposal networks, which

extracted multi-frame features to predict the pattern of object motion. By calculating the spatial changes, (Kang et al., 2016) (Kang et al., 2018) utilized the spatiotemporal proposals for final detection task. The results from (Kang et al., 2016) (Kang et al., 2018) demonstrated the state-of-the-art performance in object detection on videos.

All the above methods, however, focus on spare pixel-level propagation approach. This approach would encounter difficulties when objects dramatically changes along with their appearance of scales in the video. With the inaccurate motion estimation, the pixel-level propagation approach(Zhu et al., 2017b)(Zhu et al., 2017a) may encounter difficulties to produce desirable results. In addition, propagating pixel features from (only) few designated frames may be incapable to facilitate the final detection task if an object instance appears shorter than the propagation range. In our work, we intend to calibrate features of interests at pixel and instance level. We consider object motion in our calibration, thus we can accurately aggregate features of interest from consecutive frames instead of using spare frames as a reference. Comparing to pixel-feature propagation, our approach has more robustness to appearance variations in video frames.

3 TECHNICAL APPROACH

In this section, we present the details of our technical approach which includes the model design, inference of the model, and model architectures. Model design introduces the implementation of our model; inference of the model explains the prediction procedures of the trained model; and model architecture articulates the sub-networks of the overall deep neural network.

3.1 Model Design

The core concept of the MFCN is to calibrate imagery features at pixel level and instance level along with motion from frames $t-i$ to the current frame t to improve the detection results. Following the popular architectures of CNN for detection task(Ren et al., 2015)(Liu et al., 2016), our model has feature extraction sub-networks $\mathcal{N}_{ext}$ to produce a number of intermediate feature maps and object detection sub-networks $\mathcal{N}_{det}$ to process the extracted features maps. When camera sensor produces instant video frames I_t, $t = 1,...,\infty$, feature extraction sub-networks $\mathcal{N}_{ext}$ process the input frame and produce a number of intermediate feature maps, as $f_t = \mathcal{N}_{ext}(I_t)$. The feature maps are fed into object detection sub-networks $\mathcal{N}_{det}$, as $d_t = \mathcal{N}_{det}(I_t)$. The output d_t includes position-sensitive score maps d_t and the proposals of the input frame.

MFCN includes three contributing components: 1. pixel-feature calibration; 2. instance-feature calibration; and 3. Adaptive weight. For pixel-feature calibration, $\mathcal{N}_{ext}$ constantly receives a single frame input, and generate the intermediate feature map. We employ motion estimation sub-network to estimate motion tendency of feature maps across frames. Based on the motion estimation, we employ a bi-linear wrapping function $\mathcal{W}(\cdot,\cdot)$ to wrap pixel features from previous frames as $f_{t-i\to t}$. Meanwhile, we adopt a fully convolutional network $\varepsilon(\cdot)$ to compute the embedding features from $t-i$ to t for similarity measurement which calculates the adaptive weight for each feature maps from $t-i$. Once we have adaptive weight, we aggregate each feature map from f_{t-i} to f_t and calibrate the feature of the current frame as $f_{t-final}$.

Next, for instance-feature calibration, we use object detection sub-network to obtain position-sensitive score maps and instance proposals from frame $t-i$ to frame t. Since each proposal has ground truth information, we use a spatial-change regression network $R(\cdot)$ to estimate the movement of each proposal from the corresponding locations at frame $t-i$ to frame t. With the estimated proposal movement, we calibrate each instance feature accordingly from frame $t-i$ to current frame t. Once the calibration is done, we average the aggregated instance features from $t-i$ to obtain the final result for frame t. The procedures for pixel-feature calibration, adaptive weight calculation, and instance-feature calibration are elaborated below.

3.1.1 Pixel-feature Calibration

We model the pixel-level calibration from a previous frame I_{t-i} to current frame I_t with motion estimation. We denote $\mathcal{F}(\cdot,\cdot)$ as the motion estimation function. $\mathcal{F}(I_{t-i},I_t)$ calculates the spatial changes from frame I_t to I_{t-i}. We leverage the motion estimation to have an interference of location p_{t-i} in the previous frame $t-i$ to the location p_t in the current frame t. Let Δp be the estimated distance between p_{t-i} and p_t, and thus $\Delta p = \mathcal{F}(I_{t-i},I_t)(p_t)$. The calibrated feature maps define as:

$$f_{t-i\to t}(p_t) = \sum_q G(q_{t-i}, p_{t-i}+\Delta p) f_{t-i}(q_{t-i}). \quad (1)$$

In (2), q enumerates all spatial locations at p_{t-i} on feature map f_{t-i}, and $G(\cdot)$ denotes bi-linear interpolation kernel, which has two one-dimensional kernels

as:

$$G(q,p+\Delta p) = g(q_x, p_x+\Delta p_x)\cdot g(q_y, p_y+\Delta p_y). \quad (2)$$

Then, the calibrated feature maps from previous frames are wrapped into the current frame accordingly. The wrapping process is as the follow:

$$f_{t-i\to t} = \mathcal{W}(f_{t-i}, \mathcal{F}(I_{t-i}, I_t)). \quad (3)$$

In (3), f_{t-i} is the feature map produced by $\mathcal{N}_{ext}$ at frame $t-i$, $f_{t-i\to t}$ denotes the wrapped features from time $t-i$ to time t, and $\mathcal{W}(\cdot,\cdot)$ is a bi-linear wrapping function which is applied on each feature map of all channels across frames. Through this feature wrapping process, the current frame aggregates multiple feature maps from previous frames. To process the aggregated features, we average these features to update the feature of the current frame:

$$f_{t-final} = \sum_{j=t-i}^{t} \omega_{j\to t} f_{j\to t} \quad (4)$$

In (4), $\omega_{j\to t}$ is the adaptive weight which decides contribution of each wrapping features, $f_{j\to t}$ is the wrapping features, $f_{t-final}$ is the calibrated feature for the current frame, and i specifies calibration range from the previous frames to the current frame ($i = 3$ by default).

3.1.2 Adaptive Weight Calculation

The adaptive weight indicates the importance of each wrapped feature from previous frame to the current frame. At location p_{t-i}, if feature $f_{t-i}(p_{t-i})$ is close to the feature $f_t(p_t)$ at the current frame, a larger weight is assigned to it. On the contrary, a smaller weight is assigned. To calculate the adaptive weight, we employ the cosine similarity metric(Liu et al., 2016) to measure the similarity between the previous frame $t-i$ and the current frame. In implementation, we apply a three-layer fully convolutional network $\varepsilon(\cdot)$ to $f_{t-i\to t}$ and f_t, which projects embedding features from the similarity measurement. The measurement defines as:

$$f^s_{t-i\to t}, f^s_t = \varepsilon(f_{t-i\to t}, f_t) \quad (5)$$

In (5), $f^s_{t-i\to t}$ and f^s_t are embedding features from similarity measurement. Thus, the adaptive weight defines as:

$$\omega_{t-i\to t} = exp(\frac{f^s_{t-i\to t}\cdot f^s_t}{|f^s_{t-i\to t}||f^s_t|}). \quad (6)$$

The adaptive weight $w_{t-i\to t}$ is normalized for every spatial location over previous frames from I_{t-i} to I_t.

3.1.3 Instance-feature Calibration

After updating the feature map for the current frame, we feed $f_{t-final}$ into the detection sub-network $\mathcal{N}_{det}$ and obtain improved detection result d_t for the current frame t. The instance-feature calibration is conducted on calibrating features at instance level across frames. The final detection result at frame t is calculated by aggregating each calibrated instance features with proposal movements from previous frames.

To compute individual proposal movements from frame $t-i$ to t, we utilize the RoI pooling operation to predict the pooled features $m^n_{t-i\to t}$ of the n^{th} proposal r^n_{t-i} at location $(x^n_t, y^n_t, w^n_t, h^n_t)$ which is in the current frame t:

$$m^n_{t-i\to t} = \phi(\mathcal{F}(I_{t-i}, I_t), (x^n_t, y^n_t, w^n_t, h^n_t)). \quad (7)$$

In (7), $\phi(\cdot,\cdot)$ is the RoI pooling operation from (Girshick, 2015) and $\mathcal{F}(I_{t-i}, I_t)$ is the motion estimation function from pixel-feature calibration. Using the RoI pooling operation, we convert the features in each region of interest from proposal into a small feature map.

After the RoI pooling operation, we build a spatial-change regression network $R(\cdot)$ to predict the movements of proposals between the frame $t-i$ and t based on the $m^n_{t-i\to t}$. $R(\cdot)$ defines as:

$$(\Delta^n_{x_{t-i\to t}}, \Delta^n_{y_{t-i\to t}}, \Delta^n_{w_{t-i\to t}}, \Delta^n_{h_{t-i\to t}}) = R(m^n_{t-i\to t}). \quad (8)$$

In (8), $(\Delta^n_{x_{t-i\to t}}, \Delta^n_{y_{t-i\to t}}, \Delta^n_{w_{t-i\to t}}, \Delta^n_{h_{t-i\to t}})$ is the spatial changes which defines the movement of each proposal between frame $t-i$ and from t.

In our study, if the proposal overlaps more than 0.8 in intersection-over-union (IoU), the regression instance is considered to be a positive proposal with ground truth in movement. Only the positive proposals are learnt across frames. In the same vein, we recursively produce calibrated ground-truth of each instances with spatial changes to current frame t.

After we obtain the spatial changes of individual proposal above, we can calibrate each proposal from previous frames. The calibrated proposal defines as:

$$\begin{aligned} x^n_{t-i\to t} &= \Delta^n_{x_{t-i\to t}} \times \omega^n_t + x^n_t \\ y^n_{t-i\to t} &= \Delta^n_{y_{t-i\to t}} \times h^n_t + y^n_t \\ \omega^n_{t-i\to t} &= exp(\Delta^n_{\omega_{t-i\to t}}) \times \omega^n_t \\ h^n_{t-i\to t} &= exp(\Delta^n_{h_{t-i\to t}}) \times h^n_t. \end{aligned} \quad (9)$$

Finally, we aggregate the calibrated proposals from $t-i$ to current frame t and employ the average operation to obtain the final detection result. The final detection for n^{th} proposal defines as below:

$$d^n_{t-final} = \frac{\sum_{j=t-i}^{t} \psi(d_j, (x^n_{j\to t}, y^n_{j\to t}, w^n_{j\to t}, h^n_{j\to t})}{i+1}. \quad (10)$$

Algorithm 1: Inference Process of MFCN.

1: **input**: frame$\{I_t\}$ ◁ Image sensor produce input
2: $f_t = \mathcal{N}_{ext}(I_t)$ ◁ Extract feature maps of the input frame
3: **for** $j = t - i$ **to** t **do** ◁ Initiate pixel feature buffer
4: $f_{j\to t} = \mathcal{W}(f_j, \mathcal{F}(I_j, I_t))$ ◁ Wrap frame j to frame t
5: $f^s_{j\to t}, f^s_t = \varepsilon(f_{j\to t}, f_t)$ ◁ Calculate embedding features
6: $\omega_{j\to t} = exp(\frac{f^s_{j\to t} \cdot f^s_t}{|f^s_{j\to t}||f^s_t|})$ ◁ Calculate weight for aggregation
7: **end for**
8: $f_{t-final} = \sum_{j=t-i}^{t} \omega_{j\to t} f_{j\to t}$ ◁ Calibrate pixel features for frame t
9: $d_t = \mathcal{N}_{det}(f_{t-final})$ ◁ Produce score maps for frame t
10: **for** $j = t - i$ **to** t **do** ◁ Initiate instance feature buffer
11: **for** $i = 0$ **to** n **do** ◁ Initiate the calibration of all the n proposals from score maps
12: $(x^i_j, y^i_j, w^i_j, h^i_j) = d^i_j$ ◁ i^{th} proposal from score maps d_j
13: $m^i_{j\to t} = \phi(\mathcal{F}(I_j, I_t), (x^i_t, y^i_t, w^i_t, h^i_t))$. ◁ Predict the pooled features of the i^{th} proposal
14: $(\Delta^i_{x_{j\to t}}, \Delta^i_{y_{j\to t}}, \Delta^i_{w_{j\to t}}, \Delta^i_{h_{j\to t}}) = R(m^i_{j\to t})$ ◁ Predict the proposal movement
15: $(x^i_j, y^i_j, \omega^i_j, h^i_j) = (\Delta^i_{x_{j\to t}} \times \omega^i_t + x^i_t, ...)$ ◁ Calibrate the i^{th} proposal
16: **end for**
17: **end for**
18: $d^n_{t-final} = \frac{\sum_{j=t-i}^{t} \psi(d_j, (x^n_{j\to t}, y^n_{j\to t}, w^n_{j\to t}, h^n_{j\to t})}{i+1}$ ◁ Calibrate instance features for frame t
19: **output**: detection result $\{d^n_{t-final}\}$

In (10), d_j denotes the position-sensitive score maps from previous frames, ψ is a position-sensitive pooling operation from(Dai et al., 2016), and i the calibration range ($i = 3$ by default).

3.2 Inference Details

We summarize the MFCN inference process in Algorithm 1. The camera sensor is constantly producing image frame I_t as input. In the pixel-feature calibration process, the feature extraction sub-networks $\mathcal{N}_{ext}$ first process each input frame I_t and produce a number of intermediate feature map f_t (L2 in Algorithm 1). Based on the designated feature calibration range i, feature maps from frame $t - i$ to frame t are wrapped to update the current frame t (L4 in Algorithm 1). Meanwhile, we apply a three-layer fully convolutional network to extract the embedded features and we use them to conduct the similarity measurement to compute the adaptive weight (L5 to L6 in Algorithm 1). Pixel feature at the current frame t is calibrated by aggregating the wrapped features with adaptive weight (L8 in Algorithm 1). After this process, we calculate the proposal movements and calibrate each proposal accordingly between frame $t - i$ to frame t (L9 to L17 in Algorithm 1). Once we have the proposal movement, we adopt a position-sensitive pooling operation to process the calibrated proposals and the associated score maps. The final result $d^n_{t-final}$ is computed based on the average operation of the instance features between frame $t - i$ and t.

In this section, we introduce the experiment setup, ablation study, comparison with the State-of-the-art models, architecture evaluation, and calibration range evaluation. Collectively, the experiment design and results demonstrate the effectiveness of our model.

4 EVALUATION AND RESULTS

4.1 Experiment Setup

We evaluate the proposed model on the KITTI dataset(Geiger et al., 2013) and ImageNet VID dataset(Russakovsky et al., 2015). For KITTI, we adopt video sequences from the raw data. Note that we simplify KITTI classes into car and person for training and evaluation. For ImageNet VID dataset, we purposefully select videos focusing on car and person (on bicycle and motorcycle). We employ ImageNet VID dataset because it includes more challenging vehicular instances with motions.

In our study, all the modules described above are trained end-to-end. Experiments are performed on a workstation with 8 NVIDIA GTX 1080Ti GPUs

and Intel Core i7-4790 CPU. For training on KITTI dataset, we employ SGD optimizer with the batch size of 8 on 8GPUs. 20K iterations are performed with the learning rates of 10^{-3} and decay rate of 10^{-4}. For training on ImageNet dataset, we also employ SGD optimizer with the batch size of 8 on 8GPUs. 30K iterations are performed with the learning rates of 10^{-4} and decay rate of 10^{-5}. For feature extraction and motion estimation sub-networks, we adopt a pre-trained weight from ImageNet classification for ResNet101 and a pre-trained weight from the Flying Chair dataset (Dosovitskiy et al., 2015) for FlowNet. In the evaluation, we follow the protocol in (Geiger et al., 2013) and utilize the mean average precision (mAP) as the evaluation metric.

4.2 Ablation Study

We first conduct an ablation study to evaluate the improvements of the proposed model from a strong single-frame baseline and the pixel-feature aggregation approach. The results are listed in Table 1.

Method (a) is the single-frame baseline. We adopt the state-of-the-art R-FCN (He et al., 2016) with ResNet101 (Dai et al., 2016) for the baseline. Method (a) achieves 77.1% and 75.7% mAP on KITTI dataset and ImageNet dataset respectively.

Method (b) incorporates pixel-feature aggregation in addition to the baseline detector. We adopt FlowNet2(Ilg et al., 2017) to estimate the motion of movements. Compared to method (a), the mAP for method (b) increases 7% and 5.9% on KITTI and ImageNet respectively. Accordingly, the average precision for car and person also increases from the baseline detection. The results indicate that pixel-feature aggregation is helpful for promoting detection accuracy from the baseline detector.

Method (c) includes both pixel-feature aggregation and instance-feature aggregation but not adaptive weight. For method (c), we employ average operation for pixel-feature calibration instead of using the adaptive weight. Compared to the previous two methods, method (c) achieves higher mAP as well as the average precision (AP) in car and person on the two datasets. The mAP for method (c) is 8.9% and 1.9% higher than method (a) and (b) respectively on KITTI. The mAP for method (c) is 8.25% and 2.35% higher than method (a) and (b) respectively on ImageNet. By adopting instance feature aggregations, we observe improvements in detection accuracy from the two previous methods.

Method (d) is the proposed model which includes all the contributing modules. Compared to method (c), we add the adaptive weight for detection. On both KITTI and ImageNet dataset, method (c) achieves the highest mAP as well as the average precision (AP) in car and person compared to its counterparts. The mAP for method (d) is 10.3%, 1.4%, and 3.3% higher than the previous methods respectively on KITTI. In addition, the mAP for method (d) is 9.9%, 1.65%, and 4% higher than its counterparts respectively on ImageNet. By adopting pixel and instance feature calibration along with the adaptive weight, we achieve the best performance in detection accuracy.

The major challenges for video detection include motion blur, video defocus, truncation, occlusion, and rare pose of an instance. When encountering these problems, the baseline detector generally has incorrect inference or struggles to generate an inference; the pixel-feature aggregation approach frequently generates a loose bounding box which is not accurate enough in evaluation; while the proposed model outperforms its counterparts in such circumstance.

4.3 Comparison with the State-of-the-Art Models

We further evaluate our model by comparing some of the state-of-the-art models. We select four leading models from KITTI benchmark report (Geiger et al., 2013) whose code are publicly available. We implement the selected models and our model on the same workstation in order to have fair comparison.

Table 2 summarizes the performance of each model in detection. Based on the results, our model achieves highest mAP compared to its counterparts. For car class, our model achieves 1.13% higher than the best model in this category; while for person class,

Table 1: The Validation Results from Ablation Study.

KITTI				
Methods	a	b	c	d
Baseline	√	√	√	√
Pixel-feature calibration		√	√	√
Instance-feature calibration			√	√
Adpative Weight				√
AP(%) for Person	73.3	78.5$_{\uparrow 5.2}$	80.9$_{\uparrow 7.6}$	**82.9$_{\uparrow 9.6}$**
AP(%) for Car	80.9	89.7$_{\uparrow 8.8}$	91.1$_{\uparrow 10.2}$	**91.9$_{\uparrow 11.0}$**
mAP(%)	77.1	84.1$_{\uparrow 7.0}$	86.0$_{\uparrow 8.9}$	**87.4$_{\uparrow 10.3}$**
ImageNet				
Methods	a	b	c	d
Baseline	√	√	√	√
Pixel-feature calibration		√	√	√
Instance-feature calibration			√	√
Adpative Weight				√
AP(%) for Person	76.3	79.5$_{\uparrow 3.2}$	80.1$_{\uparrow 3.8}$	**81.7$_{\uparrow 5.4}$**
AP(%) for Car	75.1	83.7$_{\uparrow 8.6}$	87.8$_{\uparrow 12.7}$	**89.5$_{\uparrow 14.4}$**
mAP(%)	75.7	81.6$_{\uparrow 5.9}$	83.95$_{\uparrow 8.25}$	**85.6$_{\uparrow 9.9}$**

Table 2: Comparison of KITTI Benchmark.

Model	MFCN	TuSimple	RRC	sensekitti	F-PointNet
Car	91.90%	90.77%[1]	86.86%[2]	86.62%[3]	84.82%[4]
Person	82.90%	80.26%[2]	78.08%[3]	70.13%[4]	81.48%[1]
mAP(%)	87.40%	85.52%[1]	82.47%[3]	78.38%[4]	83.15%[2]

Table 3: Calibration Range Evaluation.

Aggregation Range	1	3 (default)	5	7	9	11
mAP(%)	82.1	87.4	88.9	89.2	89.3	89.3
Runtime(fps)	36.2	35.8	25.2	17.7	9.3	5.7

our model leads the top model by 1.42% in accuracy. The results indicate that the proposed model has the best overall performance among the leading models from KITTI benchmark competition. Hence we argue that our proposed approach is competitive with the existing best approaches.

4.4 Calibration Range Evaluation

For our model, calibration range is a contributing parameter for the performance of the final detection because it decides the amount of information for propagation and computation. An excessive aggregation range may cause a un-affordable computation cost while an insufficient aggregation range may optimize the detection results. Hence, we want to study the impact of aggregation range on speed-accuracy correlation.

The results are summarized in Table 3. Notice that the detection accuracy increases at a modest rate with the calibration range increasing from 3 to 11. Considering the accuracy-runtime tradeoff, the model with the default aggregation range has the best overall performance. The runtimes for calibration range between 1 to 5 range 36.2 to 25.2 fps. The inference speed reaches the frame rate of a live-stream video input which generally has 25 to 30 frames per second. The results indicate that our model is capable to process live-stream video inputs for detection purpose.

Above results provide some useful clues for practical applications. However, MFCN is only tested on two datasets, thus the results may be more heuristic than general. We participate to explore the model more thoroughly in the future.

5 CONCLUSION AND FUTURE WORK

In this study, we propose a motion-assist calibration network (MFCN) for video object detection, which is based on end-to-end learning approach. Motion estimation is the main tool we use to explore the temporal coherence of features. Multi-frame features with spatial changes are calibrated in a principled way. We conducted a large number of experiments which indicates that our model surpasses the existing state-of-the-art model in both KITTI and Cityscapes benchmarks. The results show that the motion modeling feature calibration improves feature representation and facilitates final detection in the video.

However, video object detection still has a lot of room for improvement. For safe autonomous driving, proper testing should reach a higher level. Despite this, the accuracy we achieve is still some distance from the target. We believe that a more efficient and accurate estimation network and object detection network may be beneficial to the improvement of video object detection. These unresolved issues may be instructive for some future work.

REFERENCES

Chen, X., Ma, H., Wan, J., Li, B., and Xia, T. (2017). Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1907–1915.

Dai, J., Li, Y., He, K., and Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387.

Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., and Brox, T. (2015). Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766.

Du, Y., Yuan, C., Li, B., Hu, W., and Maybank, S. (2017). Spatio-temporal self-organizing map deep network for dynamic object detection from videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5475–5484.

Feichtenhofer, C., Pinz, A., and Zisserman, A. (2017). Detect to track and track to detect. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3038–3046.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

Han, W., Khorrami, P., Paine, T. L., Ramachandran, P., Babaeizadeh, M., Shi, H., Li, J., Yan, S., and Huang, T. S. (2016). Seq-nms for video object detection. *arXiv preprint arXiv:1602.08465*.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual

recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., and Brox, T. (2017). Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2462–2470.

Kang, K., Li, H., Yan, J., Zeng, X., Yang, B., Xiao, T., Zhang, C., Wang, Z., Wang, R., Wang, X., et al. (2018). T-cnn: Tubelets with convolutional neural networks for object detection from videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(10):2896–2907.

Kang, K., Ouyang, W., Li, H., and Wang, X. (2016). Object detection from video tubelets with convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 817–825.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Lee, B., Erdenee, E., Jin, S., Nam, M. Y., Jung, Y. G., and Rhee, P. K. (2016). Multi-class multi-object tracking using changing point detection. In *European Conference on Computer Vision*, pages 68–83. Springer.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017a). Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017b). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Liu, D., Wang, Y., Chen, T., and Matson, E. T. (2019). Application of color filter and k-means clustering filter fusion in lane detection for self-driving car. In *Proceedings of The Third IEEE International Conference on Robotic Computing*.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer.

Ma, C., Huang, J.-B., Yang, X., and Yang, M.-H. (2015). Hierarchical convolutional features for visual tracking. In *Proceedings of the IEEE international conference on computer vision*, pages 3074–3082.

Ramanagopal, M. S., Anderson, C., Vasudevan, R., and Johnson-Roberson, M. (2018). Failing to learn: autonomously identifying perception failures for self-driving cars. *IEEE Robotics and Automation Letters*, 3(4):3860–3867.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

Ren, S., He, K., Girshick, R., Zhang, X., and Sun, J. (2017). Object detection networks on convolutional feature maps. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1476–1481.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, 104(2):154–171.

Wang, Y., Liu, D., Jeon, H., Chu, Z., and Matson, E. (2019). End-to-end learning approach for autonomous driving: A convolutional neural network model. In *Proceedings of the 11th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART*, pages 833–839. INSTICC, SciTePress.

Zeng, X., Ouyang, W., Yang, B., Yan, J., and Wang, X. (2016). Gated bi-directional cnn for object detection. In *European Conference on Computer Vision*, pages 354–369. Springer.

Zhang, X., Zhou, X., Lin, M., and Sun, J. (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856.

Zhu, X., Dai, J., Yuan, L., and Wei, Y. (2018). Towards high performance video object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7210–7218.

Zhu, X., Wang, Y., Dai, J., Yuan, L., and Wei, Y. (2017a). Flow-guided feature aggregation for video object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 408–417.

Zhu, X., Xiong, Y., Dai, J., Yuan, L., and Wei, Y. (2017b). Deep feature flow for video recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2349–2358.

Zitnick, C. L. and Dollár, P. (2014). Edge boxes: Locating object proposals from edges. In *European conference on computer vision*, pages 391–405. Springer.