Journal of Computing and Electronic Information Management
ISSN: 2413-1660 | Vol. 20, No. 3, 2026

MSDA-SparseInst: Real-Time Instance Segmentation via Multi-Scale Feature Fusion and Dual-Branch Attention

Yinghao Chen

Department of Computer Science and Engineering, Chongqing University of Technology, Chongqing 400054, China

Abstract: Real-time instance segmentation is crucial for urban road-scene understanding, where accurate pixel-level perception is required under complex backgrounds, occlusion, and large scale variation. However, existing efficient methods often struggle to balance segmentation accuracy and inference speed, especially for small distant objects and densely distributed instances. To address this issue, this paper proposes MSDA-SparseInst, a real-time instance segmentation framework based on SparseInst. Specifically, an improved backbone is adopted to enhance feature extraction, a Multi-scale Dilated Feature Aggregation (MDFA) module is introduced to strengthen cross-scale contextual modeling, and a lightweight dual-branch attention strategy composed of GCSA and GGCA is designed to refine decoder features. Experimental results on the Cityscapes validation set show that the proposed method achieves 21.8 AP, 43.4 AP50, and 18.1 AP75 at 30.7 FPS, improving the baseline SparseInst by 3.0 AP while maintaining real-time performance. The results demonstrate that MSDA-SparseInst provides a better trade-off between segmentation accuracy and efficiency for urban road-scene instance segmentation.

Keywords: Real-time instance segmentation; Urban road scenes; SparseInst; Multi-scale feature fusion; Attention mechanism.

1. Introduction

Instance segmentation aims to simultaneously recognize object categories and generate pixel-level masks for individual instances, and has become a fundamental task in computer vision. In urban road scenes, this capability is particularly important because intelligent transportation systems and autonomous driving require not only accurate object recognition, but also precise delineation of nearby and distant traffic participants under complex backgrounds. The Cityscapes benchmark, which focuses on high-resolution urban street scenes, has further promoted research on scene understanding in realistic driving environments and highlighted the practical importance of accurate and efficient instance segmentation for road-scene perception [1].

With the rapid development of deep convolutional networks, instance segmentation methods have achieved remarkable progress. Early representative frameworks such as Mask R-CNN adopt a two-stage pipeline that first localizes objects and then predicts masks on region features, leading to strong segmentation accuracy and high-quality object delineation [2]. However, such region-based methods usually rely on RoI operations and relatively heavy detection-and-segmentation pipelines, which limit their deployment in real-time applications. To improve efficiency, a series of one-stage or real-time methods have been proposed, including YOLACT [3], CenterMask [4], SOLOv2 [5], and CondInst [6], which reduce the computational burden by simplifying mask generation or eliminating costly region-wise processing. More recently, SparseInst introduced sparse instance activation maps and a fully convolutional, NMS-free framework, demonstrating that real-time instance segmentation can be achieved with a favorable balance between speed and accuracy [7].

Despite these advances, real-time instance segmentation in urban road scenes remains highly challenging. Compared with generic natural-image benchmarks, street-scene data contain dense traffic objects, frequent occlusions, large scale variations, and ambiguous boundaries, especially for distant or partially truncated targets. These challenges make it difficult for lightweight models to maintain both segmentation completeness and boundary quality. In particular, although SparseInst significantly improves efficiency through sparse predictions and single-level decoding, the compact design may still suffer from insufficient multi-scale contextual modeling and limited feature discrimination in complex scenes, especially when small objects, overlapping instances, and background interference appear simultaneously. Therefore, further enhancing cross-scale representation and instance-aware feature refinement under real-time constraints remains an important research problem.

To address the above issues, this paper proposes MSDA-SparseInst, a real-time instance segmentation framework based on SparseInst for urban road scenes. Without changing the sparse prediction paradigm, we enhance the model from three aspects. First, an improved backbone is introduced to strengthen geometric modeling and feature extraction while preserving computational efficiency. Second, a Multi-scale Dilated Feature Aggregation (MDFA) module is incorporated into the encoder to enrich receptive fields and improve multi-scale context representation before feature fusion. Third, to enhance the quality of decoder features, we design a dual-branch attention strategy consisting of a Global Channel-Spatial Attention (GCSA) module and a Global Grouped Coordinate Attention (GGCA) module, which strengthen informative channels, salient spatial regions, and direction-aware contextual cues with limited overhead. In this way, the proposed method improves the segmentation of small, distant, and densely distributed objects while retaining the efficient and end-to-end characteristics of SparseInst.

Extensive experiments on the Cityscapes validation set demonstrate the effectiveness of the proposed method. Compared with the baseline SparseInst, MSDA-SparseInst improves mask AP from 18.8 to 21.8 while maintaining real-time inference speed, achieving 30.7 FPS on a single RTX 3090 GPU. The improvement is also reflected under stricter localization criteria, indicating that the proposed

enhancements are beneficial not only for overall segmentation accuracy but also for mask quality and instance separation in complex road scenes.

The main contributions of this paper can be summarized as follows. We propose MSDA-SparseInst, a real-time instance segmentation framework for urban road scenes that enhances SparseInst with stronger multi-scale representation and a dual-branch attention mechanism while preserving its sparse and efficient design. Specifically, an improved backbone and an MDFA module are introduced to strengthen feature extraction and multi-scale contextual aggregation, thereby improving the representation capability for objects with large scale variations. In addition, a lightweight dual-branch attention scheme, consisting of GCSA and GGCA, is designed to refine decoder features and improve robustness against background interference, occlusion, and boundary ambiguity. Extensive experiments on the Cityscapes validation set demonstrate that the proposed method achieves a 3.0 AP improvement over the baseline SparseInst while maintaining real-time performance, indicating a better trade-off between segmentation accuracy and inference speed.

2. Related Work

2.1. Instance Segmentation Methods

Instance segmentation aims to assign a semantic category and a pixel-level mask to each individual object in an image. As a fundamental task in computer vision, it has attracted extensive attention due to its broad applications in autonomous driving, robotics, and intelligent surveillance. Existing methods can generally be grouped into region-based approaches and one-stage or detector-free approaches according to their object representation and prediction pipeline.

Region-based methods usually follow a top-down framework, in which objects are first localized and then segmented based on region features. Among them, Mask R-CNN is one of the most representative approaches. It extends Faster R-CNN by introducing a parallel mask prediction branch on top of Region of Interest (RoI) features and achieves strong segmentation accuracy with a clear and effective architecture [8]. Building upon this paradigm, many subsequent studies further improve mask quality by refining object localization, enhancing region feature extraction, or introducing cascade structures [9]. Although region-based methods often achieve high accuracy and produce fine object contours, they generally rely on proposal generation and RoI-based operations, which introduce considerable computational overhead and limit their efficiency in real-time scenarios.

To improve inference efficiency, a series of one-stage and detector-free methods have been proposed in recent years. YOLACT decomposes instance segmentation into two parallel processes, namely prototype mask generation and instance-specific coefficient prediction, thereby achieving fast inference with a simple fully convolutional framework. CondInst further removes explicit RoI operations by generating instance-aware dynamic convolution parameters for mask prediction, which improves flexibility and efficiency. CenterMask builds upon an anchor-free detector and introduces an additional mask branch to realize real-time instance segmentation in a more efficient one-stage manner. Different from detection-based pipelines, SOLOv2 formulates instance segmentation from a location-based perspective and dynamically generates mask kernels for each instance without relying on bounding box refinement, showing that direct and fast instance segmentation is also feasible.

Despite the remarkable progress of these methods, the trade-off between accuracy and efficiency remains a core issue in instance segmentation. Region-based methods usually provide stronger mask quality but suffer from higher latency, whereas many efficient one-stage methods improve speed at the cost of limited contextual modeling or insufficient robustness in complex scenes. This problem becomes more evident in dense and scale-varying environments, where accurate instance separation and fine-grained mask prediction are both required. Therefore, designing an instance segmentation framework that can better balance segmentation quality and computational efficiency is still an important research topic.

2.2. Real-Time Instance Segmentation Based on Sparse Predictions

Real-time instance segmentation has attracted increasing attention because many practical applications, such as autonomous driving and robotic perception, require both accurate mask prediction and low-latency inference. However, conventional real-time methods often still rely on dense object representations, multi-level prediction, or post-processing steps such as non-maximum suppression, which introduce redundant computation and limit deployment efficiency. In this context, sparse prediction has emerged as an appealing direction for reducing inference overhead while preserving competitive segmentation performance. SparseInst is a representative method along this line, as it reformulates real-time instance segmentation with a sparse and fully convolutional framework rather than relying on dense anchors, center priors, or RoI-based operations.

The core idea of SparseInst is to represent objects by a sparse set of instance activation maps (IAMs), which highlight informative regions for each instance and directly aggregate instance-level features from the whole image. Compared with region-based and center-based representations, IAMs can better emphasize discriminative pixels, suppress irrelevant background responses, and avoid the extra region extraction operations required by RoI-based pipelines. To make such flexible activation-based representations trainable, SparseInst formulates label assignment as a bipartite matching problem and adopts Hungarian matching to realize one-to-one prediction. Benefiting from sparse predictions, single-level decoding, compact architecture, and simple post-processing without NMS or sorting, SparseInst achieves a favorable speed–accuracy trade-off and provides an efficient baseline for real-time instance segmentation.

Following this trend, subsequent studies further explored efficient sparse or lightweight instance prediction paradigms. For example, FastInst showed that query-based instance segmentation can also be adapted to real-time settings through lighter decoders and more efficient query updates [10], while RTMDet-Ins demonstrated that strong real-time detection frameworks can be extended to instance segmentation with only lightweight kernel and mask-feature branches [11]. These studies suggest that recent real-time instance segmentation research is moving toward simpler architectures, lighter prediction heads, and lower post-processing cost. Nevertheless, for complex urban road scenes, existing

efficient methods still face difficulty in handling large scale variation, dense object distribution, and ambiguous boundaries, which leaves room for further improvement on top of the SparseInst framework.

2.3. Multi-Scale Feature Fusion and Attention Mechanisms for Urban Road Scenes

Urban road-scene instance segmentation is particularly challenging because traffic objects usually exhibit large scale variation, frequent occlusion, complex background interference, and ambiguous boundaries. In practical driving environments, nearby vehicles often occupy large image regions, whereas distant vehicles, pedestrians, and riders appear as small-scale targets with sparse and weak visual cues. In addition, illumination variation, weather changes, and partial truncation further increase the difficulty of accurate instance discrimination. As a result, real-time segmentation models often struggle to maintain both mask completeness and boundary precision in urban scenes, especially for distant or densely distributed instances.

To address scale variation, multi-scale feature fusion has become a widely adopted strategy in dense prediction tasks. Feature Pyramid Network (FPN) improves multi-scale representation by combining high-level semantic features with low-level detailed features through a top-down pathway and lateral connections [12]. Building upon FPN, PANet shortens the information propagation path by introducing bottom-up path aggregation [13], which enhances localization cues from shallow layers. EfficientDet further proposes BiFPN to achieve more efficient bidirectional feature fusion with lightweight repeated connections [14]. These studies show that effective cross-scale interaction is crucial for handling objects of different sizes. For urban road scenes in particular, stronger multi-scale fusion is beneficial not only for large nearby objects, but also for small distant instances whose features are easily submerged by dominant foreground regions or cluttered backgrounds.

Besides feature fusion, attention mechanisms have also been extensively used to improve feature representation by emphasizing informative regions and suppressing irrelevant responses. Representative channel-attention methods such as SE explicitly model channel interdependencies to recalibrate convolutional features [15], while CBAM further combines channel and spatial attention to refine features in two complementary dimensions [16]. Coordinate Attention embeds positional information into channel attention and has shown effectiveness in lightweight networks by enhancing spatially selective responses with limited overhead [17]. In road-scene perception, such mechanisms are especially valuable because they help the model focus on salient target regions, preserve weak cues of small or distant objects, and reduce interference from shadows, background textures, and overlapping instances. Therefore, combining multi-scale fusion with lightweight attention has become an effective way to improve robustness and segmentation quality in complex traffic environments.

Although existing studies have demonstrated the value of multi-scale fusion and attention enhancement, efficient urban road-scene instance segmentation still faces a difficult trade-off between richer feature modeling and real-time inference. In particular, for SparseInst-based frameworks, it remains meaningful to strengthen cross-scale contextual aggregation and refine decoder features with lightweight attention modules, so as to better handle small distant objects, dense distributions, and boundary ambiguity without sacrificing efficiency. This observation provides the main motivation for the method proposed in this paper.

3. Method

3.1. Overview

As illustrated in Fig. 1, MSDA-SparseInst is a unified framework for real-time instance segmentation. It consists of three main components: an improved backbone for feature extraction, an instance context encoder for cross-scale feature aggregation, and an IAM-based decoder for instance recognition and mask prediction [7].

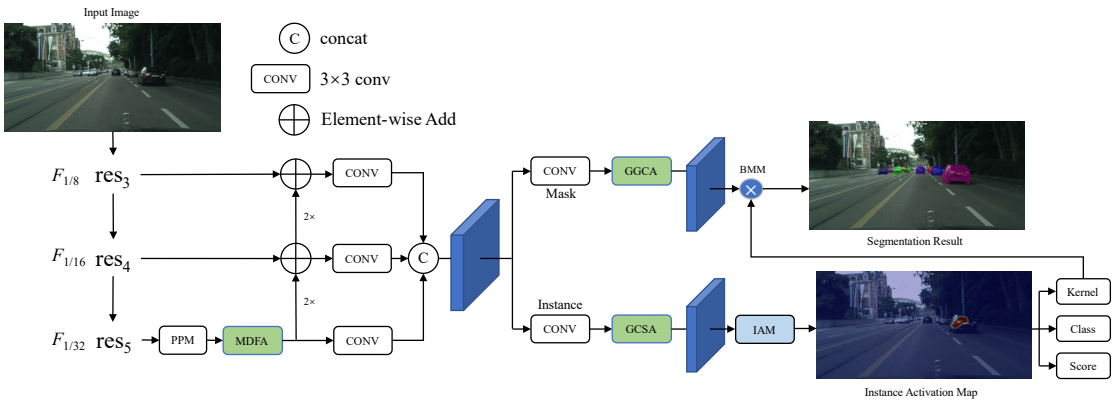

Fig. 1 Overall architecture of MSDA-SparseInst

Given an input image, the backbone first extracts multi-level convolutional features. In our implementation, a ResNet-50-based backbone is adopted [18], where a deep stem and average downsampling are used to strengthen low-level feature extraction [19], and the later stages are enhanced with deformable bottlenecks to better model geometric variations with limited overhead [20].

The extracted multi-level features are then fed into the instance context encoder, which aggregates contextual information and fuses cross-scale cues to produce a compact single-level feature map for subsequent prediction. Following the sparse and efficient design of SparseInst, the encoder preserves rich semantics while maintaining computational efficiency. In addition, to strengthen multi-receptive-field context under real-time constraints, an MDFA block is inserted on the top feature after pyramid pooling and before top-down fusion [21].

Finally, the IAM-based decoder contains an instance branch and a mask branch. The instance branch predicts instance-level outputs, including category scores, objectness

scores, and instance-specific mask kernels, while the mask branch generates shared mask features for segmentation [7]. To improve robustness in complex scenes, we further enhance the decoder with a dual-branch attention design, where GCSA is introduced in the instance branch and GGCA is incorporated into the mask branch. The final instance masks are obtained by applying the predicted kernels to the shared mask features.

3.2. Improved Backbone

We adopt a ResNet-50-style backbone and introduce lightweight modifications to enhance geometric modeling while maintaining real-time efficiency [18]. Specifically, we use a deep stem with average downsampling to strengthen low-level feature extraction [19]. Furthermore, we enable deformable modeling in the res4 and res5 stages of the backbone to better handle geometric variations in complex scenes [20]. An overview of the improved backbone and the modified bottleneck design is illustrated in Fig. 2.

Deformable Auxiliary Residual Bottleneck (DARB). For the stages with deformable modeling enabled, we replace the standard 3×3 convolution in the ResNet bottleneck with DCNv2 [18, 20]. An additional prediction branch is used to generate learnable offsets together with modulation scalars[20], allowing the convolution to perform adaptive spatial sampling and response reweighting. The main residual pathway follows the original bottleneck design: it first reduces channels with a 1×1 convolution [18], then applies a 3×3 deformable convolution, and finally expands channels with another 1×1 convolution. This design improves spatial modeling flexibility with limited additional overhead.

Auxiliary residual branch. To further improve representation capacity with minimal complexity, we introduce an auxiliary residual branch that projects the block input to the output channels using a 1×1 convolution followed by normalization. The output of the modified bottleneck can be written as $y = F_{\mathrm{dcn}}(x) + S(x) + \beta \cdot A(x)$, where $F_{\mathrm{dcn}}(\cdot)$ denotes the main deformable transformation, $S(\cdot)$ is the shortcut connection, $A(\cdot)$ is the auxiliary projection branch, and β is a learnable scaling factor. This design preserves the original residual structure while providing an additional lightweight information path for deep feature enhancement.

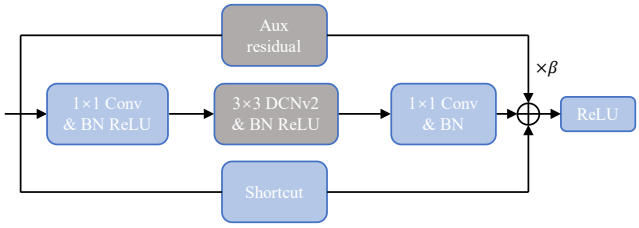

Fig. 2 Structure of the Deformable Auxiliary Residual Bottleneck (DARB) in the improved backbone.

3.3. Multi-scale Dilated Feature Aggregation Module

Although our encoder employs pyramid pooling and top-down fusion to construct a unified feature representation, the top-level feature map can still be insufficient for small or distant instances due to its low spatial resolution and limited local details. To enhance contextual modeling before cross-scale propagation, we introduce a lightweight Multi-scale Dilated Feature Aggregation (MDFA) module along the top-level path, as shown in Fig. 3.

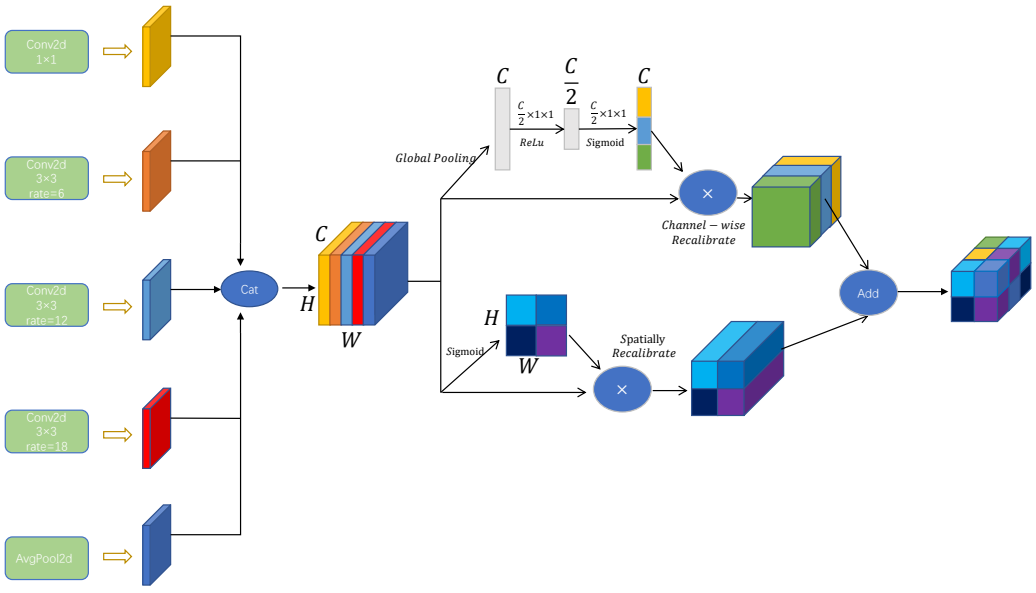

Fig. 3 Architecture of the Multi-scale Dilated Feature Aggregation (MDFA) module

In instance segmentation, objects often exhibit large variations in scale and shape. A fixed receptive field may over-emphasize either local details or global context. Dilated convolutions provide an effective way to enlarge the receptive field without reducing feature-map resolution [22], which is beneficial for capturing multi-scale cues while preserving spatial details. Therefore, MDFA is designed to aggregate complementary contextual information using parallel dilated convolutions with different dilation rates together with a global context branch [22].

Module design. Given an input feature map $X \in R^{C \times H \times W}$, MDFA extracts multi-scale context through five parallel branches, including: (i) a 1×1 convolution branch for local channel mixing, (ii) three 3×3 dilated convolution branches with dilation rates 6, 12, and 18 to capture medium-to-large context at multiple scales, and (iii) a global average pooling branch to encode image-level context [21, 22], followed by a 1×1 convolution and bilinear upsampling to restore the spatial resolution. The outputs of the five branches are first concatenated along the channel dimension to form a fused representation. The concatenated feature is then recalibrated by channel-wise and spatial attention, where the learned

channel and spatial weights are applied through element-wise multiplication. The recalibrated feature is finally compressed and integrated by a 1×1 convolution to obtain the aggregated output, with batch normalization and ReLU applied where appropriate.

Lightweight residual integration. To keep the computational overhead low, we apply MDFA via a lightweight residual wrapper. Specifically, we first reduce the channel dimension to a hidden size $C_h \leq C$, feed the reduced feature into MDFA, and then restore the channel dimension to the original size using a 1×1 convolution followed by batch normalization and a ReLU activation. Finally, the projected feature is added back to the original input. This design helps retain the original feature pathway while enabling the network to progressively exploit multi-scale contextual information.

Insertion point. In our implementation, the MDFA residual-lite block is applied immediately after PPM on the top-level (res5) feature, and the subsequent top-down fusion and final concatenation fusion remain unchanged.

3.4. Global Channel-Spatial Attention Module

To enhance the instance-branch features in the single-level decoder and suppress background interference, we introduce a lightweight Global Channel-Spatial Attention module (GCSA), as shown in Fig. 4. GCSA performs channel reweighting, channel shuffle, and spatial refinement sequentially [16, 23], so that instance-related cues can be strengthened with limited computational overhead.

Channel attention. Given an input feature map $X \in R^{B\times C\times H\times W}$, we first reshape the spatial dimensions and then use a lightweight two-layer MLP to learn channel dependencies [15], producing channel-wise gates for feature recalibration. The gates are activated by a sigmoid function and applied to the input feature via element-wise multiplication, so that informative semantic channels can be emphasized while less relevant responses are suppressed.

Channel shuffle. After channel reweighting, we perform channel shuffle with G groups to encourage cross-group feature interaction and alleviate group-wise feature isolation [23]. This operation improves channel mixing and provides more discriminative representations for subsequent spatial refinement.

Spatial attention. We further refine the feature map with a spatial attention branch implemented by a lightweight 7×7 convolutional bottleneck [23]. The output is activated by a sigmoid function and multiplied with the input feature, highlighting informative spatial regions while suppressing irrelevant background responses.

For clarity, the computation of GCSA can be summarized as a sequential process of channel reweighting, channel shuffle, and spatial refinement: $X_c = X \cdot \mathrm{Sigmoid}(\mathrm{MLP}(X))$; $X_s = \mathrm{Shuffle}(X_c)$; $Y = \mathrm{GCSA}(X) = X_s \cdot \mathrm{Sigmoid}(f_s(X_s))$ where $\cdot$ denotes element-wise multiplication. $\mathrm{MLP}(\cdot)$ is a lightweight two-layer perceptron applied along the channel dimension at each spatial location. $\mathrm{Shuffle}(\cdot)$ denotes channel shuffle with G groups, and $f_s(\cdot)$ denotes the spatial attention bottleneck implemented by a 7×7 convolutional bottleneck.

Residual-gated integration. To preserve the original feature pathway and stabilize optimization, we integrate GCSA into the decoder in a residual-gated manner, so that the module starts from an identity-like mapping and is gradually activated during training. In our implementation, GCSA is inserted only into the instance branch, before the instance activation prediction layer.

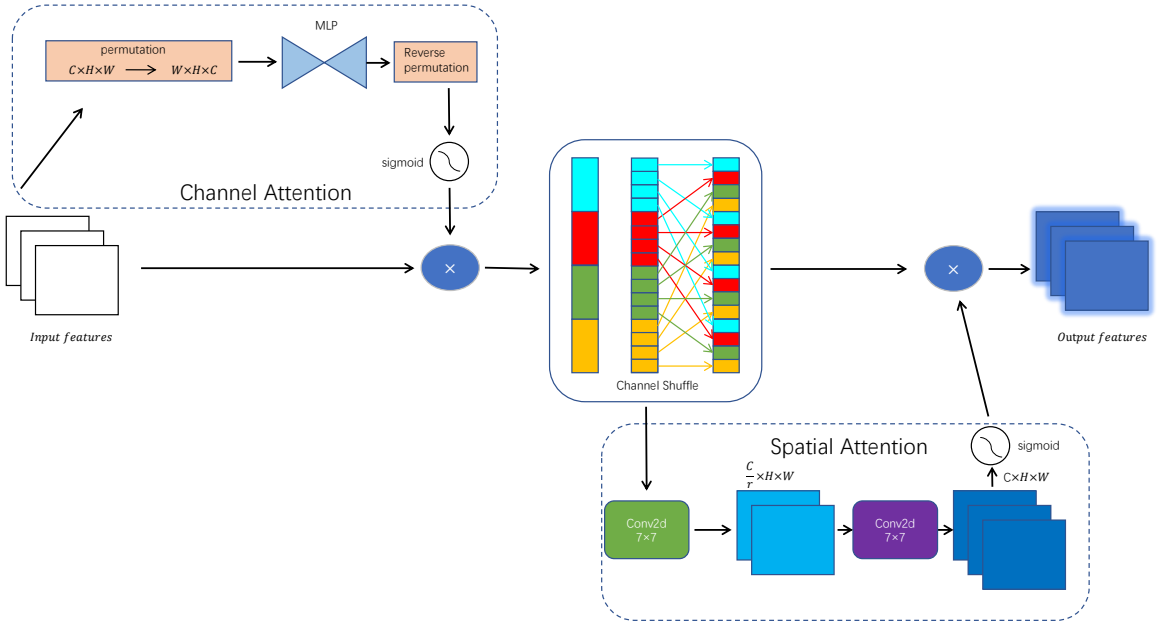

Fig. 4 Architecture of the Global Channel-Spatial Attention (GCSA) module

3.5. Global Grouped Coordinate Attention Module

We propose Global Grouped Coordinate Attention (GGCA) to enhance mask features by modeling long-range dependencies along spatial coordinates with lightweight grouped computation. Given an input feature map X, GGCA performs group-wise coordinate attention and uses a shared cross-dimensional transformation to generate direction-aware attention weights efficiently [17]. The overall architecture of GGCA is illustrated in Fig. 5.

Grouped feature partition. We first divide channels into groups, where each group contains $C_g = C/$ channels. The feature map is reshaped into grouped features $X^{(g)} \in R^{C_g\times H\times W}$. This grouped partition reduces computation while preserving within-group channel interactions.

Direction-aware pooling. For each group, we extract coordinate descriptors along the height and width directions. Specifically, for the height direction, features are aggregated along the width dimension using both adaptive average pooling and adaptive max pooling to obtain two complementary descriptors. Similarly, for the width direction, features are aggregated along the height dimension with average pooling and max pooling [16, 17]. These directional descriptors are then fused to encode long-range contextual cues along one spatial axis while retaining positional information along the other axis. Their formulations can be written as:

$$Z_h^{avg} = AvgPool_w\left(X^{(g)}\right) \in \mathbb{R}^{C_g \times H \times 1}, \quad (1)$$

$$Z_w^{avg} = AvgPool_h\left(X^{(g)}\right) \in \mathbb{R}^{C_g \times 1 \times W}, \quad (2)$$

where $AvgPool_w(\cdot)$ and $MaxPool_w(\cdot)$ denote pooling along the width dimension, and $AvgPool_h(\cdot)$ and $MaxPool_h(\cdot)$ denote pooling along the height dimension.

Shared bottleneck and attention generation. To generate attention maps efficiently, we adopt a shared cross-dimensional bottleneck with reduction ratio r for the directional descriptors. This shared transformation compresses and restores within-group channel information, and is followed by normalization and nonlinear activation where appropriate. The height- and width-direction attentions are then computed as:

$$A_h = \sigma\left(f\left(Z_h^{avg}\right) + f\left(Z_h^{max}\right)\right), \quad A_w = \sigma\left(f\left(Z_w^{avg}\right) + f\left(Z_w^{max}\right)\right) \quad (3)$$

$$Y^{(g)} = X^{(g)} \odot A_h \odot A_w, \quad Y = Reshape^{-1}\left(Y^{(g)}\right) \quad (4)$$

where $\sigma(\cdot)$ denotes the sigmoid function, and A_h and A_w correspond to the attention maps for the height and width directions, respectively.

Integration into the mask branch. Inside GGCA, the grouped feature is recalibrated by the two coordinate attentions through element-wise multiplication and then reshaped back to the original channel arrangement. The enhancement process can be written as: where $\odot$ denotes element-wise multiplication, and A_h and A_w are broadcast along the corresponding dimensions to align with $X^{(g)}$. In the decoder, GGCA is inserted after the stacked 3×3 mask convolutions and before the final projection layer, so that the mask branch can exploit direction-aware contextual cues without changing the overall decoder structure.

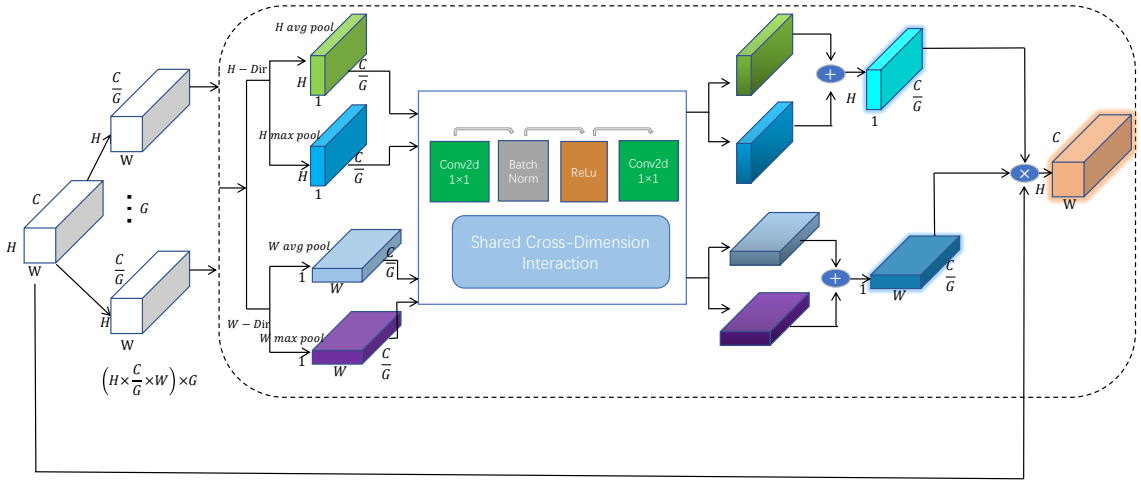

Fig. 5 Architecture of the Global Grouped Coordinate Attention (GGCA) Module

4. Experiments

4.1. Experimental Environment

All experiments were conducted on a Linux server. Training and inference were performed in a single-GPU setting using an NVIDIA GeForce RTX 3090 with 24 GB of memory under CUDA 11.3. The instance segmentation framework was implemented with Python 3.8.10 and PyTorch 1.10.0, and built upon Detectron2. The Cityscapes instance segmentation dataset was adopted in this study [1], where the training set was used for model training and the validation set was used for performance evaluation. Following the 8-class road-scene instance segmentation setting, the target categories include person, rider, car, truck, bus, train, motorcycle, and bicycle.

During training, AdamW was used as the optimizer, with an initial learning rate of 2 × 10^-4, a weight decay of 0.05, a batch size of 8, and a maximum of 120,000 iterations. The learning rate was scheduled using WarmupCosineLR, with 1,000 warm-up iterations and a warm-up factor of 0.02. Gradient clipping was further applied with a clipping threshold of 2.0. For data augmentation, only random horizontal flipping and multi-scale training were adopted. Specifically, during training, the short side of the input image was randomly selected from {512, 576, 640, 704, 768}, while the maximum image size was limited to 1536. Random cropping was also enabled, with a crop size of 512 × 1024.

Model performance was evaluated using the AP metrics commonly adopted in instance segmentation, while inference speed in FPS was additionally reported to assess both segmentation accuracy and real-time performance. All subsequent ablation and comparison experiments were conducted under this unified evaluation protocol.

4.2. Datasets and Implementation Details

Dataset and Evaluation Metrics. Our experiments are conducted on the Cityscapes dataset with fine annotations, which consists of 2,975 images for training, 500 for validation and 1,525 for testing [1]. All models are trained on the training set and evaluated on the validation set. For instance segmentation, we report the COCO-style mask AP [24], including AP averaged over IoU thresholds from 0.50 to 0.95, as well as AP50 and AP75. We measure the inference speed on a single NVIDIA GeForce RTX 3090 GPU. The reported

latency is computed by timing the model forward pass (including mask generation and resizing) with single-scale input (shorter side 768 and max size 1536) and batch size 1, and averaged after the first 100 warm-up iterations. TensorRT and FP16 are not used for acceleration.

Implementation Details. Our method is built upon SparseInst and implemented with Detectron2 on PyTorch [25]. All experiments are conducted on a single NVIDIA GeForce RTX 3090 GPU with CUDA 11.3 and PyTorch 1.10.0. We adopt the AdamW optimizer with an initial learning rate of 2×10^-4 and a weight decay of 0.05 [26]. Models are trained for 120k iterations with a total batch size of 8, and we use a warm-up cosine learning-rate schedule with 1k warm-up iterations. The backbone is initialized with ImageNet-pretrained weights and is based on ResNet-50 with deformable convolution [18, 20, 27], where batch normalization layers are frozen. We adopt random horizontal flip and scale jitter in training. The shorter side of images is randomly sampled from {512, 576, 640, 704, 768} pixels, while the longer side is less than or equal to 1536. We further apply random cropping with an absolute-range size of 512×1024. Unless otherwise specified, we evaluate the speed and accuracy with the shorter side set to 768. We set N=100 instance queries and adopt G=4 groups in the grouped modules. Loss weights for classification, dice, pixel-wise mask loss, and objectness are set to 2.0, 2.0, 5.0, and 1.0, respectively, and the mask binarization threshold is set to 0.45.

4.3. Main Results

We evaluate instance segmentation on the Cityscapes validation set using COCO-style mask average precision. We report AP, AP50, and AP75, and also measure inference speed in frames per second. For fair comparison, all methods in Table 1 are evaluated on the same dataset split and under the same evaluation protocol, while their inference speeds are measured on a single NVIDIA GeForce RTX 3090 GPU using the timing protocol described in Section 4.2.

Table 1 shows that MSDA-SparseInst achieves 21.8 AP, 43.4 AP50, and 18.1 AP75 at 30.7 FPS. The baseline SparseInst achieves 18.8 AP at 34.4 FPS, so our method improves AP by 3.0 points with only a small reduction in speed. The gain is also consistent under stricter overlap criteria, where AP75 increases from 15.5 to 18.1. MSDA-SparseInst outperforms FastInst and RTMDet-Ins in both accuracy and speed. FastInst achieves 21.0 AP at 26.0 FPS and RTMDet-Ins achieves 21.1 AP at 24.4 FPS, while our method achieves higher AP and runs at 30.7 FPS. YOLOv11-seg runs faster at 33.1 FPS, but its accuracy is notably lower at 19.3 AP. These results indicate that MSDA-SparseInst provides the highest accuracy among all compared methods while maintaining real-time inference speed.

Table 1. Comparison with state-of-the-art real-time instance segmentation methods on the Cityscapes validation set

Method	FPS	AP	AP50	AP75
YOLACT	26.2	19.0	37.9	15.8
CondInst	22.2	19.4	38.7	16.1
SOLOv2	24.1	19.8	39.5	16.5
FastInst	26.0	21.0	42.0	17.5
RTMDet-Ins	24.4	21.1	42.2	17.6
YOLOv11-seg	33.1	19.3	38.5	16.1
YOLOv8-seg	30.4	17.2	34.4	14.4
CenterMask	24.2	18.9	37.7	15.7
SparseInst(Baseline)	34.4	18.8	37.5	15.5
MSDA-SparseInst	30.7	21.8	43.4	18.1

4.4. Ablation Experiments

We conduct ablation experiments on the Cityscapes validation set to analyze the impact of key design choices and the contribution of each proposed component. Unless otherwise stated, all models are trained and evaluated using the same settings as in Section 4.2, and FPS is measured following the same timing protocol.

We first study the design of the improved backbone. As illustrated in Fig. 6, we introduce an auxiliary residual branch and evaluate three candidate insertion positions within the bottleneck block, denoted as P1, P2, and P3. The results are reported in Table 2. All three variants outperform the baseline, and inserting the auxiliary residual at P3 achieves the best overall performance, reaching 20.1 AP together with 39.2 AP50 and 16.5 AP75. Therefore, we adopt P3 as the default setting of the improved backbone in the remaining experiments.

Next, we evaluate the proposed design in a progressive manner by incrementally adding the improved backbone, MDFA, GCSA, and GGCA to the baseline SparseInst. Table 3 summarizes the results. Using the improved backbone increases AP from 18.8 to 20.1, while FPS decreases from 34.4 to 32.3. Adding MDFA further improves AP to 20.7 with only a negligible speed drop. Introducing GCSA on top of the previous configuration brings AP to 21.3 at 31.0 FPS. Finally, incorporating GGCA yields the best performance, achieving 21.8 AP at 30.7 FPS. Overall, the progressive integration of the proposed components yields consistent performance gains, and the full model improves AP by 3.0 points over the baseline while maintaining real-time inference speed.

To further isolate the effect of each proposed module, we additionally compare single-module variants built upon the DARB-enhanced backbone. As shown in Table 3, adding MDFA improves the performance to 20.7 AP and 41.7 AP50 at 32.1 FPS, indicating that richer multi-scale contextual modeling is particularly beneficial for improving matching quality under relatively loose localization thresholds. Replacing MDFA with GCSA yields 20.8 AP and 39.7 AP50 at 31.2 FPS, suggesting that instance-branch attention more directly enhances global dependency modeling and feature selection, although with a slightly larger speed reduction. Using GGCA instead gives 20.5 AP and 40.2 AP50 at 32.0 FPS, showing that direction-aware mask refinement can improve segmentation quality with relatively low overhead. Overall, MDFA, GCSA, and GGCA provide complementary gains, and their combination leads to the best overall performance.

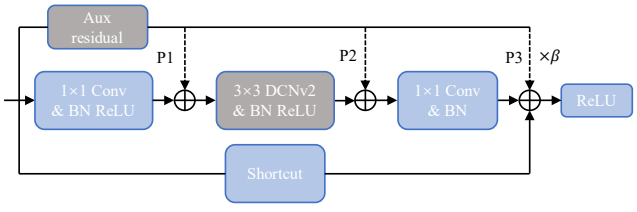

Fig. 6 Auxiliary residual injection in the bottleneck block and candidate insertion positions P1–P3

Table 2. Effect of auxiliary residual insertion position in the improved backbone on the Cityscapes validation set

Insertion position	AP	AP50	AP75
Baseline	18.8	37.5	15.5
P1	19.7	38.4	16.3
P2	19.8	38.7	16.5
P3	20.1	39.2	16.5

Table 3. Component-wise ablation of MSDA-SparseInst on the Cityscapes validation set

Variant	AP	AP50	FPS
Baseline	18.8	37.5	34.4
Baseline+DARB	20.1	39.2	32.3
Baseline+DARB+MDFA	20.7	41.7	32.1
Baseline+DARB+GCSA	20.8	39.7	31.2
Baseline+DARB+GGCA	20.5	40.2	32.0
Baseline+DARB+MDFA +GCSA	21.3	41.9	31.0
Baseline+DARB+MDFA +GCSA+GGCA	21.8	43.4	30.7

4.5. Visualizations

Fig. 7 presents visual comparisons between the baseline SparseInst and the proposed MSDA-SparseInst on the Cityscapes validation set. The first column shows the input images, the second column shows the segmentation results of SparseInst, and the third column shows the results of MSDA-SparseInst.

As shown in the first row, when road scenes contain objects at multiple scales, such as nearby vehicles together with distant small targets, SparseInst tends to yield insufficient detection results and incomplete mask coverage for distant vehicles. In contrast, MSDA-SparseInst preserves the integrity of nearby vehicle segmentation while providing more stable predictions for distant objects. As a result, the overall segmentation is more coherent and the detection of small-scale distant targets becomes more sufficient, demonstrating better adaptability to multi-scale scenarios.

In the second row, where a large-sized target appears together with adjacent vehicles, the contour fitting of SparseInst is less stable, and some local regions exhibit coarse boundaries or inaccurate mask coverage. By comparison, MSDA-SparseInst restores vehicle shapes more accurately and achieves better alignment between the predicted masks and the true object contours. This helps preserve structural details more completely and improves the geometric consistency of instance segmentation.

Fig. 7 Visual comparison between SparseInst and MSDA-SparseInst on the Cityscapes validation set

In the third row, when targets are located near image boundaries or when truncated objects and background interference are present, SparseInst is more likely to generate incomplete masks and less clean boundaries for some instances, while also being relatively conservative in detecting and segmenting small roadside objects. MSDA-SparseInst produces clearer instance boundaries in such complex regions and maintains better recognition and segmentation performance for vehicles near the image borders as well as small roadside targets, thereby reducing missed detections and contour loss in boundary areas.

As shown in the fourth row, in scenes with a relatively large number of vehicles, object instances are often densely distributed and partially occluded. Under such conditions, the instance separation ability of SparseInst is affected in local regions, and the boundaries between neighboring vehicles are not always well distinguished. In comparison, MSDA-SparseInst achieves clearer separation among densely distributed vehicles and preserves the independence of adjacent instances more reliably, leading to more consistent segmentation results in crowded scenes.

In the fifth row, when the road scene is relatively open but the targets are far away and have comparatively low contrast, SparseInst tends to produce unstable responses or insufficient mask coverage for distant objects. MSDA-SparseInst still maintains better segmentation completeness for long-range vehicles in this case, and the instance predictions over near-, mid-, and far-range targets appear more continuous, indicating stronger robustness.

Overall, the visual comparisons on the Cityscapes validation set show that, compared with the baseline SparseInst, MSDA-SparseInst delivers more stable and accurate segmentation results in terms of multi-scale object representation, contour fitting, boundary-region handling,

and instance separation in crowded scenes. These improvements lead to better mask completeness and higher boundary quality.

5. Conclusion

This paper presented MSDA-SparseInst, a real-time instance segmentation framework for urban road scenes based on SparseInst. By introducing an improved backbone, the MDFA module, and a lightweight dual-branch attention design with GCSA and GGCA, the proposed method strengthens multi-scale contextual modeling and decoder feature refinement without changing the sparse prediction paradigm. As a result, the model achieves better robustness in challenging scenarios involving small distant objects, dense distributions, and ambiguous boundaries.

Experimental results on the Cityscapes validation set showed that MSDA-SparseInst achieved 21.8 AP at 30.7 FPS, improving the baseline SparseInst by 3.0 AP while retaining real-time performance. The ablation and visualization results further confirmed the effectiveness of each proposed component and the superiority of the full model in segmentation completeness, boundary quality, and instance separation. These results indicate that MSDA-SparseInst offers an effective solution for balancing accuracy and efficiency in real-time urban road-scene instance segmentation. Future work will focus on improving the performance on extremely small or heavily occluded targets and exploring more deployment-oriented lightweight optimizations.

References

[1] Cordts M, Omran M, Ramos S, et al. The cityscapes dataset for semantic urban scene understanding[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 3213-3223.

[2] He K, Gkioxari G, Dollár P, et al. Mask r-cnn[C]//Proceedings of the IEEE international conference on computer vision. 2017: 2961-2969.

[3] Bolya D, Zhou C, Xiao F, et al. Yolact: Real-time instance segmentation[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2019: 9157-9166.

[4] Lee Y, Park J. Centermask: Real-time anchor-free instance segmentation[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 13906-13915.

[5] Wang X, Zhang R, Kong T, et al. Solov2: Dynamic and fast instance segmentation[J]. Advances in Neural information processing systems, 2020, 33: 17721-17732.

[6] Tian Z, Shen C, Chen H. Conditional convolutions for instance segmentation[C]//European conference on computer vision. Cham: Springer International Publishing, 2020: 282-298.

[7] Cheng T, Wang X, Chen S, et al. Sparse instance activation for real-time instance segmentation[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022: 4433-4442.

[8] Ren S, He K, Girshick R, et al. Faster R-CNN: Towards real-time object detection with region proposal networks[J]. IEEE transactions on pattern analysis and machine intelligence, 2016, 39(6): 1137-1149.

[9] Cai Z, Vasconcelos N. Cascade r-cnn: Delving into high quality object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 6154-6162.

[10] He J, Li P, Geng Y, et al. Fastinst: A simple query-based model for real-time instance segmentation[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2023: 23663-23672.

[11] Lyu C, Zhang W, Huang H, et al. Rtmdet: An empirical study of designing real-time object detectors[J]. arXiv preprint arXiv:2212.07784, 2022.

[12] Lin T Y, Dollár P, Girshick R, et al. Feature pyramid networks for object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 2117-2125.

[13] Liu S, Qi L, Qin H, et al. Path aggregation network for instance segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 8759-8768.

[14] Tan M, Pang R, Le Q V. Efficientdet: Scalable and efficient object detection[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 10781-10790.

[15] Hu J, Shen L, Sun G. Squeeze-and-excitation networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 7132-7141.

[16] Woo S, Park J, Lee J Y, et al. Cbam: Convolutional block attention module[C]//Proceedings of the European conference on computer vision (ECCV). 2018: 3-19.

[17] Hou Q, Zhou D, Feng J. Coordinate attention for efficient mobile network design[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021: 13713-13722.

[18] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[19] He T, Zhang Z, Zhang H, et al. Bag of tricks for image classification with convolutional neural networks[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 558-567.

[20] Zhu X, Hu H, Lin S, et al. Deformable convnets v2: More deformable, better results[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 9308-9316.

[21] Zhao H, Shi J, Qi X, et al. Pyramid scene parsing network[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 2881-2890.

[22] Chen L C, Papandreou G, Schroff F, et al. Rethinking atrous convolution for semantic image segmentation[J]. arXiv preprint arXiv:1706.05587, 2017.

[23] Zhang X, Zhou X, Lin M, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 6848-6856.

[24] Lin T Y, Maire M, Belongie S, et al. Microsoft coco: Common objects in context[C]//European conference on computer vision. Cham: Springer International Publishing, 2014: 740-755.

[25] Paszke A, Gross S, Massa F, et al. Pytorch: An imperative style, high-performance deep learning library[J]. Advances in neural information processing systems, 2019, 32.

[26] Loshchilov I, Hutter F. Decoupled weight decay regularization[J]. arXiv preprint arXiv:1711.05101, 2017.

[27] Russakovsky O, Deng J, Su H, et al. Imagenet large scale visual recognition challenge[J]. International journal of computer vision, 2015, 115(3): 211-252.