OSFormer: One-Stage Camouflaged Instance Segmentation with Transformers

Jialun Pei[†1][0000−0002−2630−2838], Tianyang Cheng[†2][0000−0001−7910−3624], Deng-Ping Fan[*3][0000−0002−5245−7518], He Tang[2][0000−0002−8454−1407], Chuanbo Chen[2][0000−0001−8006−7851], and Luc Van Gool[3][0000−0002−3445−5711]

[1] School of Computer Science and Technology, HUST, China
[2] School of Software Engineering, HUST, China
[3] Computer Vision Lab, ETH Zurich, Switzerland

Abstract. We present **OSFormer**, the first one-stage transformer framework for camouflaged instance segmentation (CIS). OSFormer is based on two key designs. First, we design a **location-sensing transformer** (LST) to obtain the location label and instance-aware parameters by introducing the location-guided queries and the blend-convolution feed-forward network. Second, we develop a **coarse-to-fine fusion** (CFF) to merge diverse context information from the LST encoder and CNN backbone. Coupling these two components enables OSFormer to efficiently blend local features and long-range context dependencies for predicting camouflaged instances. Compared with two-stage frameworks, our OSFormer reaches 41% AP and achieves good convergence efficiency without requiring enormous training data, *i.e.*, only 3,040 samples under 60 epochs. Code link: https://github.com/PJLallen/OSFormer.

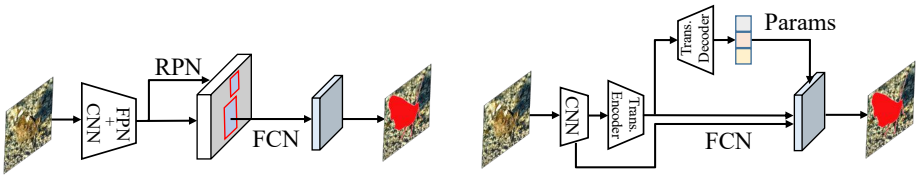
Keywords: Camouflage, instance segmentation, transformer

1 Introduction

Camouflage is a powerful and widespread means of avoiding detection or recognition that stems from biology [51]. In nature, camouflage objects have evolved a suite of concealment strategies to deceive perceptual and cognitive mechanisms of prey or predators, such as background matching, self-shadow concealment, obliterative shading, disruptive coloration, and distractive markings [11, 48]. These defensive behaviors make camouflaged object detection (COD) a very challenging task compared to generic object detection [5,32,42,44,50]. COD is dedicated to distinguishing camouflaged objects with a high degree of intrinsic similarity with backgrounds [16]. It is essential to use computer vision models to assist human visual and perceptual systems for COD, such as polyp segmentation [17,28], lung infection segmentation [18], wildlife protection, and recreational art [10].

Thanks to the build of large-scale and standard benchmarks like COD10K [16], CAMO [31], CAMO++ [30], and NC4K [37], the performance of COD has received significant progress. However, COD only separates camouflaged objects

† Equal contributions. * Corresponding author (dengpfan@gmail.com).



(a) Two-stage: Mask R-CNN (CNN+FPN) **(b) One-stage:** OSFormer (CNN+Transformer)

Fig. 1. Framework comparisons of Mask R-CNN [23] and the proposed OSFormer.

from the scene at object-level while ignoring further instance-level identification. Recently, Le *et al.* [30] presented a new camouflaged instance segmentation (CIS) benchmark and a camouflage fusion learning framework. Capturing camouflaged instances can provide more clues (*e.g.*, semantic category, the number of objects) in real-world scenarios, thus CIS is more challenging. To this end, in this paper, we focus on the CIS task.

Compared to generic instance segmentation [23], CIS needs to be performed in more complex scenarios with high feature similarity and results in class-agnostic masks. Moreover, various instances may display different camouflage strategies in a scene, and combining them may form mutual camouflage. These derived ensemble camouflages make the CIS task even more daunting. When humans gaze at a heavily camouflaged scene, the visual system will instinctively sweep across a series of local scopes throughout the whole scene to search for valuable clues [38, 45]. Inspired by this visual mechanism, we present a novel location-aware CIS approach that meticulously captures crucial information at all positions (*i.e.*, *local context*) in a *global perspective* and directly generates camouflaged instance masks (*i.e.*, *one-stage* model).

Thanks to the rise of the transformer [52] in the visual domain, we can employ self-attention and cross-attention to capture long-range dependencies and build global content-aware interactions [5]. Although the transformer model has shown strong performance on some dense prediction tasks [22, 53, 54, 60], it requires embracing large-scale training data and longer training epochs. However, there is currently only limited instance-level training data available as a brand-new downstream task. To this end, we propose a **location-sensing transformer (LST)** based on [65] to achieve faster convergence and higher performance with fewer training samples. To dynamically yield location-guided queries for each input image, we grid the multi-scale global features output from the LST encoder into a set of feature patches with varying local information. Compared to zero initialization of object queries in vanilla DETR [5], the proposed location-guided queries can lead to focus on location-specific features and interact with global features through cross-attention to gain the instance-aware embeddings. This design effectively speeds up convergence and significantly improves detecting camouflaged instances. To enhance local perception and the correlation between neighboring tokens, we introduce convolution operations into the standard feed-

forward network [52], which we term blend-convolution feed-forward network (BC-FFN). Therefore, our LST-based model can seamlessly integrate local and global context information and efficiently provide location-sensitive features for segmenting camouflaged instances.

In addition, we design a **coarse-to-fine fusion (CFF)** to integrate multi-scale low- and high-level features successively derived from ResNet [24] and LST to bring out the shared mask feature. Since the edges of the camouflaged instances are difficult to capture, a reverse edge attention (REA) module is embedded in our CFF module to enhance the sensitivity to edge features. Finally, inspired by [25], we introduce the dynamic camouflaged instance normalization (DCIN) to generate the masks by uniting the high-resolution mask feature and the instance-aware embeddings. Based on those mentioned above two novel designs, *i.e.*, LST and CFF, we provide a new one-stage framework **OSFormer** for camouflaged instance segmentation (Fig. 1). To the best of our knowledge, OSFormer is the first work to explore the transformer-based framework for the CIS task. Our **contributions** are as follows:

1. We propose **OSFormer**, the first one-stage transformer-based framework designed for the camouflage instance segmentation task. It is a flexible framework that can be trained in an end-to-end manner.
2. We present a **location-sensing transformer (LST)** to dynamically seize instance clues at different locations. Our LST contains an encoder with the blend-convolution feed-forward network to extract multi-scale global features and a decoder with the proposed location-guided queries to bring the instance-aware embeddings. The proposed LST structure converges quickly with limited *i.e.*, about 3,000 images, training data.
3. A novel **coarse-to-fine fusion (CFF)** is proposed to get the high-resolution mask by fusing multi-scale low- and high-level features from the backbone and LST block. In this module, reverse edge attention (REA) is embedded to highlight the edge information of camouflaged instances.
4. Extensive experiments show that OSFormer performs well for the challenging CIS task, **outperforming** 11 popular instance segmentation approaches by a large margin, *e.g.*, *8.5% AP improvement* on the COD10K test set.

1.1 Related Work

Since camouflage instance segmentation is a relatively new task, here, we review some works (*e.g.*, camouflaged object detection, generic instance segmentation, vision transformer) that are closest to ours.

Camouflaged Object Detection. This category of the models aims to identify objects that blend in the surrounding scene [20]. Earlier studies mainly employed low-level handcrafted contrast features and certain heuristic priors (*e.g.*, color [27], texture [2, 47] and motion boundary [41]) to build camouflaged object detection (COD) models. With the popularity of deep learning architecture and the release of large-scale pixel-level COD datasets [16, 31], the performance of COD has been improved by leaps and bounds in the past two years. Deep

learning methods [39,43,61,64] utilize CNNs to extract high-level informative features to search and locate camouflaged objects, and then design an FCN-based decoder to purify features to predict camouflaged maps. For instance, Mei *et al.* [39] presented a positioning and focus network (PFNet) to mimic the process of predation in nature. PFNet first leverages the positioning module to locate the potential targets and use the focus module to refine the ambiguous regions. Zhai *et al.* [63] adopted a mutual graph learning strategy to train the regions and edges of camouflaged objects interactively. Afterward, Lyu *et al.* [37] proposed a ranking network that simultaneously localizes, segments, and ranks concealed objects for better prediction. Recently, a novel uncertainty-guided transformer-based model proposed by Yang *et al.* [62] is designed to infer uncertain regions with Bayesian learning. The COD task ignores the instance-level predicted maps essential for actual application scenarios despite the rapid development. Thus, we are dedicated to advancing the COD task from object-level to instance-level.

Generic Instance Segmentation. Existing works can be roughly summarized as the top-down and bottom-up patterns. The former model performs a classic detect-then-segment design that first detects ROIs by bounding boxes and then segment pixel-level instances locally [49]. The typical model is Mask R-CNN [23], which extends Faster R-CNN [44] by adding a mask branch to predict instance-level masks. On this basis, Mask Scoring R-CNN [26] introduces a MaskIoU head to assess the quality of the instance mask. To enhance the feature pyramid and shorten the information flow, PANet [35] creates a bottom-up path augmentation. Furthermore, Chen *et al.* [7] proposed Hybrid Task Cascade (HTC) to interweave detection and segmentation features for joint processing. Different from the above-mentioned two-stage models, YOLACT [3] is a real-time one-stage framework that embraces two parallel tasks: producing non-local prototype masks and predicting a set of mask coefficients.

In contrast to the top-down manner, the bottom-up methods first learn instance-aware holistic embeddings and then identify each specific instance with clustering operations [8, 34]. Bai *et al.* [1] proposed an end-to-end boundary-aware deep model derived from the classical watershed transform. SSAP [21] can jointly learn the pixel-level semantic class and instance differentiating by an instance-aware pixel-pair affinity pyramid. However, the performance of previous bottom-up models is inferior to top-down models because of the suboptimal pixel grouping. To this end, Tian *et al.* [49] presented a dynamic instance-aware network that directly outputs instance masks in a fully convolutional paradigm. The simpler strategy is efficient and performs favorably against Mask R-CNN-like frameworks. Furthermore, SOLO [56,57] detects the center location of instances by semantic categories and decouples the mask prediction into the dynamic kernel feature learning. Inspired by this strategy, we design a location-aware network based on the transformer to dynamically perceive camouflaged instances.

Vision Transformer. Transformer [52] was born out of natural language processing and has been successfully extended to the field of computer vision [15]. The core idea of the encoder-decoder transformer architecture is a self-attention mechanism that builds long-range dependencies and captures global context in-

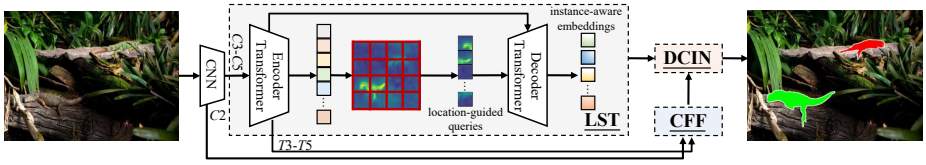


Fig. 2. OSFormer includes the location-sensing transformer (LST), coarse-to-fine fusion (CFF) module, and dynamic camouflaged instance normalization (DCIN) module.

formation from an input sequence. Recently, Carion *et al.* proposed DETR [5], which combined transformer with CNN backbone to aggregate object-related information and provided a group of object queries to output the final set of predictions. Despite DETR pioneering a novel and concise paradigm, it still suffers from the high computational cost and the slow convergence. Considering these issues, many efforts focused on how to develop a more efficient DETR architecture [12, 13, 65]. Zhu *et al.* [65] introduced a deformable attention layer embedded in the self-attention module to reduce the computational cost and training schedule. UP-DETR [13] leveraged a novel unsupervised pretext task to pre-train the transformer of DETR for accelerating convergence. However, most existing transformer models are adapted to vision tasks with many training data. Therefore, for downstream tasks with only small datasets, fully utilizing the performance of transformer is an urgent issue to be solved. To this end, we present an efficient location-sensing transformer (LST) based on the deformable DETR [65] for CIS. Specifically, rather than zero-initialized object queries in DETR [5], we design a set of dynamic location-guided queries that are generated by gridding the multi-scale global features from the LST encoder. In addition, we introduce convolutional operations in the feed-forward network to enhance inductive biases and local perception. Benefiting from the above-mention designs, the proposed transformer converges easily on the CIS task with only 3,040 training samples.

2 OSFormer

Architecture. The proposed OSFormer comprises four essential components: (1) a CNN backbone to extract object feature representation, (2) a location-sensing transformer (LST) that utilizes the global feature and location-guided queries to produce the instance-aware embeddings. (3) a coarse-to-fine fusion (CFF) to integrate multi-scale low- and high-level features and yield a high-resolution mask feature, and (4) a dynamic camouflaged instance normalization (DCIN) that is applied to predict the final instance masks. We illustrate the whole architecture in Fig. 2.

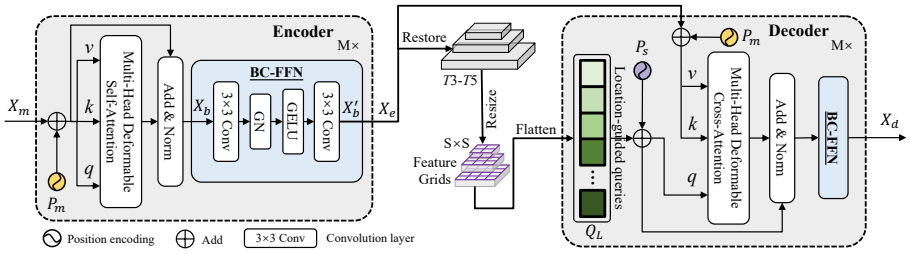


Fig. 3. Structure of our location-sensing transformer.

2.1 CNN Backbone

Given an input image $I \in \mathbb{R}^{H\times W\times 3}$, we use multi-scale features $\{Ci\}_{i=2}^{5}$ from the CNN backbone (*i.e.*, ResNet-50 [24]). To reduce the computational cost, we directly flatten and concatenate the last three feature maps $(C3, C4, C5)$ into a sequence X_m with 256 channels as input to the proposed LST encoder (Sec. 2.2). For $C2$ feature, we feed it into our CFF (Sec. 2.3) module as high-resolution low-level feature to capture more camouflaged instance cues.

2.2 Location-Sensing Transformer

Although the transformer can better extract global information by the self-attention layer, it requires large-scale training samples and high computational cost to support. Due to the limited data of CIS, our goal is to design an efficient architecture that can converge faster and achieve competitive performance. In Fig. 3, we present our location-sensing transformer (LST).

LST Encoder. Unlike DETR [5] with only a single-scale low-resolution feature input to the transformer encoder, our LST encoder receive multi-scale features X_m to obtain rich information. Following deformable self-attention layers [65] to better capture local information and enhance the correlation between neighboring tokens, we bring the convolution operations into the feed-forward network, named blend-convolution feed-forward network (BC-FFN). First, the feature vector is restored to the spatial dimension depending on the shape of Ci. Then, a convolution layer with the kernel size of 3×3 is performed to learn the inductive biases. Finally, we add a group normalization (GN) and a $GELU$ activation to form our feed-forward network. After a 3×3 convolution layer, we flatten the features back into a sequence. Compared to mix-FFN [60], our BC-FFN contains no MLP operations and residual connections. Unlike [59] that designs a convolutional token embedding at the beginning of each stage and employs depth-wise separable convolution operation in the transformer block, we only bring two convolution layers in BC-FFN. Specifically, given an input feature X_b, the process of *BC-FFN* can be formulated as:

$$X_b' = Conv^3(GELU(GN(Conv^3(X_b)))), \tag{1}$$

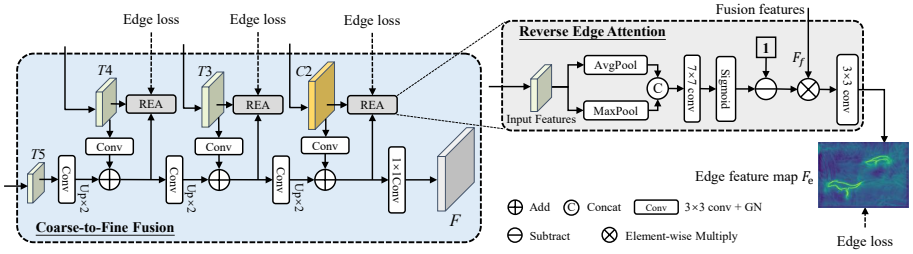


Fig. 4. Structure of our coarse-to-fine fusion.

where $Conv^3$ is a 3×3 convolution operation. Overall, a LST encoder layer is described as follows:

$$X_e = BC\text{-}FFN(LN((X_m + P_m) + MDAttn(X_m + P_m))), \tag{2}$$

where P_m is denoted as the position encodings. $MDAttn$ and LN represent multi-head deformable self-attention and layer normalization, respectively.

Location-Guided Queries. Object queries play a critical role in the transformer architecture [5], which are used as the initial input to the decoder and attain the output embeddings through the decoder layers. However, one of the reasons for the slow convergence of the vanilla DETR is that object queries are zero-initialized. To this end, we propose location-guided queries that take advantage of multi-scale feature maps $Ti, i = 3, 4, 5$ from the LST encoder[4]. It is noteworthy that each query in DETR concentrates on specific areas. Inspired by SOLO [56], we first resize the restored feature maps $T3$-$T5$ to the shape of $S_i \times S_i \times D, i = 1, 2, 3$. Then, we divide the resized features into $S_i \times S_i$ feature grids and flatten them to produce our location-guided queries $Q \in \mathbb{R}^{L \times D}, L = \sum_{i=1}^{3} S_i^2$. In this situation, the proposed location-guided queries can utilize learnable local features in different locations to optimize the initialization and efficiently aggregate the features in the camouflaged areas. Compared to the zero or random initialization [5, 65], this query strategy improves the efficiency of query updates in the transformer decoder and accelerates the training convergence. For more discussion, please refer to Sec. 3.1.

LST Decoder. The LST decoder is essential to interact with the global features produced by the LST encoder and location-guided queries for producing the instance-aware embeddings. Spatial position encoding is also added to our location-guided queries Q_L and the encoder memory X_e. After that, they are fused by the deformable cross-attention layer. Unlike the general transformer decoder, we directly use cross-attention without self-attention because the proposed queries already contain learnable global features. BC-FFN is also employed after deformable attention operations, similar to the LST encoder. Given

[4] We split and restore the X_e to the 2D representations $T3 \in \mathbb{R}^{\frac{H}{8} \times \frac{W}{8} \times D}$, $T4 \in \mathbb{R}^{\frac{H}{16} \times \frac{W}{16} \times D}$, and $T5 \in \mathbb{R}^{\frac{H}{32} \times \frac{W}{32} \times D}$.

location-guided queries Q_L, the pipeline of our LST decoder is summarized as:

$$X_d = BC\text{-}FFN(LN((Q_L + P_s) + MDCAttn((Q_L + P_s), (X_e + P_m)))), \quad (3)$$

where P_s represents the position encoding based on the feature grids. $MDCAttn$ is denoted as the multi-head deformable cross-attention operation. X_d is the output embeddings for instance-aware representation. Finally, X_d is restored to feed into the following DCIN module (Sec. 2.4) for predicting masks.

2.3 Coarse-to-Fine Fusion

As a bottom-up transformer-based model, OSFormer strives to utilize multi-level global features output from the LST encoder to result in a shared mask feature representation. To merge diverse context information, we also fuse the low-level feature $C2$ from the CNN backbone as a complement to yield a unified high-resolution feature map $F \in \mathbb{R}^{\frac{H}{4} \times \frac{W}{4} \times D}$. The detailed structure of the proposed coarse-to-fine fusion (CFF) is shown in Fig. 4. We take the multi-level features $C2, T3, T4$, and $T5$ as input for cascade fusion. Starting from $T5$ at 1/32 scale of input, a 3×3 convolution, GN, and $2\times$ bilinear upsampling are passed and added with the higher-resolution feature ($T4$ with 1/16 scale). After fusing $C2$ with a 1/4 scale, the feature proceeds through a 1×1 convolution, GN, and RELU operations to generate the mask feature F. Note that each input feature reduces the channels from 256 to 128 after the first convolution and then is increased to 256 channels at the final output.

Considering that the edge features of camouflage appear more challenging to capture, we design a reverse edge attention (REA) module embedded in CFF to supervise the edge features during the iterative process. Unlike the previous reverse attention [9, 17], our REA operates on the edge features rather than the predicted binary masks. In addition, the edge labels used for supervision are obtained by erosion of instance mask labels without any manual labeling. Inspired by the Convolutional Block Attention [58], the input features are operated by both average-pooling ($AvgPool$) and max-pooling ($MaxPool$). Then, we concatenate and forward them to a 7×7 convolution and a sigmoid function. Afterward, we reverse the attention weight and apply them to the fusion feature F_f by element-wise multiplication. Lastly, we use a 3×3 convolution to predict the edge feature. Assuming that the input feature is Ti, the whole process of each REA module can be formulated as follows:

$$F_e = Conv^3(F_f \otimes (1 - Sigmoid(Conv^7([AvgPool(Ti); MaxPool(Ti)])))), \quad (4)$$

where $Conv^7$ represents the 7×7 convolution layer, and $[;]$ denotes concatenation on the channel axis. In a word, the proposed CFF provides a shared mask feature F to feed into the DCIN to predict the final camouflaged per-instance mask.

2.4 Dynamic Camouflaged Instance Normalization

Inspired by the instance normalization operation in the style transfer domain [25, 46], we introduce a dynamic camouflaged instance normalization (DCIN) to

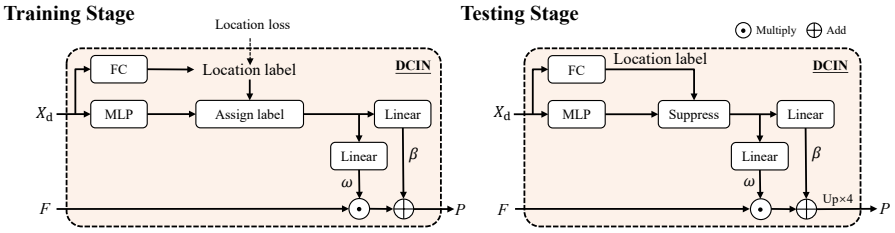


Fig. 5. Structure of our dynamic camouflaged instance normalization.

predict final masks. When DCIN receives the output embeddings $X_d \in \mathbb{R}^{S^2 \times D}$ from the LST decoder, a fully-connected layer (FC) is employed to gain the location label. In parallel, a multi-layer perceptron (MLP) is used to gain the instance-aware parameters with a size of D (*i.e.*, 256). We assign positive and negative locations according to the ground truth in the training stage. The instance-aware parameters of the positive location are applied to generate the segmentation mask. In the testing stage, we utilize the confidence value of the location label to filter (See Suppress in Fig. 5) ineffective parameters (*e.g.*, Threshold > 0.5). Subsequently, two linear layers are operated on the filtered location-aware parameters to attain affine weights $\omega \in \mathbb{R}^{N \times D}$ and biases $\beta \in \mathbb{R}^{N \times 1}$, respectively. Finally, they are used together with the shared mask feature $F \in \mathbb{R}^{\frac{H}{4} \times \frac{W}{4} \times D}$ to predict the camouflaged instances, which can be described as:

$$P = U_{\times 4}(\omega F + \beta), \tag{5}$$

where $P \in \mathbb{R}^{H \times W \times N}$ is the predicted masks. N is the number of predicted instances. $U_{\times 4}$ is an upsampling operation by a factor of 4. In the end, the Matrix NMS [57] is applied to get the final instances.

2.5 Loss Function

During the training, the total loss function can be written as:

$$L_{total} = \lambda_{edge} L_{edge} + \lambda_{loc} L_{loc} + \lambda_{mask} L_{mask}, \tag{6}$$

where L_{edge} is the edge loss to supervise edges from different levels in our CFF. The edge loss can be defined as $L_{edge} = \sum_{j=1}^{J} L_{dice}^{(j)}$, where J represents the total number of levels of edge features for supervision, which can be seen in Fig. 4. λ_{edge} is the weight for edge loss that is set to 1 by default. Since the CIS task is category-agnostic, we use the confidence of the existence of camouflages in each location (L_{loc}) compared to the classification confidence in generic instance segmentation. In addition, L_{loc} is implemented by Focal loss [32] and L_{mask} is computed by Dice loss [40] for segmentation. λ_{loc} and λ_{mask} are set to 1 and 3 respectively to balance the total loss.

3 Experiments

Datasets. As a brand-new challenging CIS task, few task-specific datasets so far. Comfortingly, Fan *et al.* contributed a COD dataset [16], namely COD10K, which simultaneously provides high-quality instance-level annotations for training CIS models. Concretely, COD10K contains 3,040 camouflaged images with instance-level labels for training and 2,026 images for testing. Recently, Le *et al.* [30] provided a larger CIS dataset called CAMO++, which includes a total of 5,500 samples with hierarchically pixel-wise annotation. Furthermore, Lyu *et al.* [37] introduced a test set with 4,121 images, called NC4K. We use the instance-level annotations in COD10K to train the proposed OSFormer and evaluate it on the COD10K and NC4K test set.
Metrics. We adopt COCO-style evaluation metrics including AP_{50}, AP_{75}, and AP scores [33] to evaluate segmentation results. In contrast to the mAP metric in instance segmentation, each camouflaged instance detected from concealed regions is class-agnostic. Therefore, we only need to consider the existence of camouflaged instances while ignoring the mean value of the category.
Technical Details. Our OSFormer is implemented in PyTorch on a single RTX 3090 GPU and trained with Stochastic Gradient Descent. For fair comparisons, we adopt wisely used ResNet-50 backbone [24] that is initialized from the pre-trained weights of ImageNet [14]. If not specially mentioned, other backbones used in our experiments are also pre-trained on ImageNet. During training, all our models are trained for 90K iterations (60 epochs) with a batch size of 2 and a base learning rate of $2.5e-4$ with a warm-up of 1K iterations. Then, the learning rate is divided by 10 at 60K and 80K, respectively. In addition, the weight decay is set to 10^{-4} and the momentum is 0.9. The input images are resized such that the size of the shortest side is from 480 to 800 while the longest side is at most 1,333. We also use the scale jittering augmentation for data augmentation. In our LST, S_1, S_2, and S_3 are set to 36, 24, and 16, respectively. Note that the dimension of the features is kept at 256 throughout the whole process of BC-FFN. We embedd a total of six encoder layers stacked sequentially. To reach better performance, we only repeat the LST decoder layer three times to aggregate camouflaged cues relevant to queries.

3.1 Ablation Studies

We conduct a series of ablation studies on the instance-level COD10K dataset [16] to validate the effectiveness of our OSFormer and determine the hyper-parameters. The ablation experiments mainly consist of the following aspects: layers of encoder and decoder in LST, number of multi-scale feature inputs, location-guided queries designs, feature fusion in the CFF module, backbone architecture, real-time settings, and contributions of different components.
Layers of Encoder and Decoder in LST. The depth of the transformer is a key factor influencing the performance and efficiency of transformer-based models. We attempt multiple combinations of different numbers of encoder and decoder layers in our LST to optimize the performance of OSFormer. As shown

in Table. 1, the first three rows indicate that three layers are insufficient to maximize the performance of OSFormer. In addition, we observe that LST is more sensitive to the encoder than the decoder. The highest value of AP is reached when the number of encoder and decoder layers are 6 and 3, respectively. When more layers are added, the accuracy is not further improved, and the inference time drops to under $14fps$. As a result, we adopt 6 encoder layers and 3 decoder layers as our default setting to balance performance and efficiency.

Number of Multi-scale Feature Inputs. We utilize multi-level features extracted from the ResNet-50 as input of our LST. To more accurately capture camouflages at different scales while maintaining model efficiency, we combine different numbers of features in the backbone, including $C3$-$C5$, $C2$-$C5$, $C3$-$C6$, and $C2$-$C6$. In Table. 2, we observe that the combination of $C3$-$C5$ achieves a strong performance with the lowest number of parameters and training memory.

Location-Guided Queries Designs. Object queries are essential in the transformer architecture for dense prediction tasks. To validate the effectiveness of our location-guided queries, we compare two typical object query designs, including zero-initialized in vanilla DETR [5] and learnable input embeddings in deformable DETR [65]. We set the number of queries uniformly to the default number of multi-scale feature grids for fair comparisons. Other settings in the proposed OSFormer remain unchanged. In a nutshell, object queries in transformer decoder comprise two parts: query features and query position embeddings. In the vanilla DETR, an all-zero matrix added by a set of learnable position embeddings is taken as object queries through the decoder to generate the corresponding output embeddings. In contrast, the deformable DETR is directly initialized by learnable embeddings as query features and is coupled with learnable position embeddings. As seen in Table. 3, our location-guided queries are significantly superior to other query designs. It illustrates that inserting supervised global features in queries is crucial to regressing different camouflage cues and locating instances efficiently. Furthermore, we compare the learning ability of three strategies. We find that our location-guided queries scheme has a faster convergence rate at the early training stage, and the final convergence is also better than the other two models. It also demonstrates that location-guided queries are efficient to exploit global features to capture camouflaged information at different locations by cross-attention mechanism.

Feature Fusion in CFF. In the proposed CFF module, the multi-scale input features directly impact the quality of the mask feature F operated by fusing. To explore the optimal fusion scheme from ResNet-50 and LST encoder, we conduct different combinations in Table. 4. Using only a single-scale feature $T2$ without

Table 1. Effect of the different number of encoder and decoder layers in our LST.

Encoder	Decoder	AP	AP_{50}	AP_{75}	FPS
1	3	37.0	68.0	35.4	**21.8**
3	1	39.2	69.1	38.5	20.0
3	3	39.4	70.2	39.3	18.8
3	6	38.9	68.6	37.9	17.2
6	3	**41.0**	**71.1**	40.8	14.5
6	6	40.6	70.3	**41.2**	13.4
9	6	40.7	70.6	40.4	11.3

Table 2. Ablations for different combinations of multi-scale features input to our LST.

Scales	Number	AP	AP_{50}	AP_{75}	#Params	Memory
$C3$-$C5$	3	**41.0**	**71.1**	**40.8**	**46.58M**	**6.4G**
$C2$-$C5$	4	39.9	70.5	38.7	46.80M	9.3G
$C3$-$C6$	4	40.8	70.6	40.9	47.39M	6.7G
$C2$-$C6$	5	40.2	69.9	40.3	47.62M	17.7G

Table 3. Comparison of different query designs on the proposed OSFormer.

Queries	AP	AP_{50}	AP_{75}
Zero-Initialized [5]	34.7	64.1	33.1
Learnable Embeddings [65]	35.0	64.8	33.2
Location-Guided Queries (**Ours**)	**41.0** $_{+6.0}$	**71.1** $_{+6.3}$	**40.8** $_{+7.6}$

Table 4. Comparison of different feature combinations input to our CFF module.

Features	AP	AP_{50}	AP_{75}
Single $T2$	38.0	69.2	36.8
$C2, C3, C4, C5$	35.4	64.3	34.6
$C2, C3, C4, T5$	40.0	69.7	40.1
$C2, C3, T4, T5$	39.5	69.9	39.0
$T2, T3, T4, T5$	40.0	70.1	40.0
$C2, T3, T4, T5$	**41.0**	**71.1**	**40.8**

Table 5. Performance of OSFormer with different backbone networks.

Backbone Networks	AP	AP_{50}	AP_{75}	FPS
ResNet-50 [24] (**Default**)	41.0	71.1	40.8	**14.5**
ResNet-101 [24]	42.0	71.3	42.8	12.9
PVTv2-B2-Li [55]	47.2	74.9	**49.8**	13.2
Swin-T [36]	**47.7**	**78.6**	49.3	12.6

multi-scale fusion is not promising. The result of the 2^{nd} row illustrates that fusing only the backbone features is inefficient. In the end, we attain the optimal results by feeding $C2, T3, T4$, and $T5$ into the CFF module. It can be explained that the features from our LST encoder have more detailed global information. Furthermore, the feature $C2$ is also required to provide some low-level features as a supplement. In addition, we visualize the features at each scale input to the CFF module and the mask feature F in Fig. 6.

Backbone Networks. In this experiment, we use different backbone networks *i.e.*, ResNet-50 [24], ResNet-101 [24], PVTv2-B2-Li [55], and Swin-T [36], to train our OSFormer. All of them are pre-trained on ImageNet [14]. From Table. 5, we observe that OSFormer only with the ResNet-50 can achieve 41% AP scores. Moreover, using a more powerful backbone can further stimulate the potential of our method that further improves to 47.7% AP.

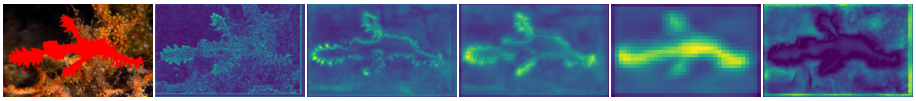


Fig. 6. Visualizations of feature maps. (a) The input image overlapped with ground-truth; (b)-(e) is the input features of CFF produced from the CNN backbone and our LST encoder; (f) is the mask feature F output from CFF.

Table 6. Models are trained on the COD10K and tested on its testing set.

Models	Backbones	AP	AP_{50}	FPS	#Params	FLOPs
OSFormer (**Default**)	ResNet-50	**41.0**	**71.1**	14.5	46.6M	324.7G
OSFormer-550	ResNet-50	36.0	65.3	**25.8**	**42.4M**	**138.7G**

Table 7. Ablation studies for different components in our OSFormer.

Encoder	LGQ	BC-FFN	CFF	REA	AP	AP_{50}	AP_{75}
	✓	✓	✓	✓	33.7	63.4	32.0
✓	✓	✓	✓	✓	34.7	64.1	33.1
✓	✓		✓	✓	37.2	67.3	35.8
✓	✓	✓			38.0	69.2	36.8
✓	✓	✓	✓		39.3	69.7	38.5
✓	✓	✓	✓	✓	**41.0**	**71.1**	**40.8**

Table 8. Quantitative comparisons with 11 representative methods.

	Methods	Backbones	Params	FLOPs	COD10K-Test AP	COD10K-Test AP_{50}	COD10K-Test AP_{75}	NC4K-Test AP	NC4K-Test AP_{50}	NC4K-Test AP_{75}
Two-Stage	Mask R-CNN [23]	ResNet-50	43.9M	186.3G	25.0	55.5	20.4	27.7	58.6	22.7
	Mask R-CNN [23]	ResNet-101	62.9M	254.5G	28.7	60.1	25.7	36.1	68.9	33.5
	MS R-CNN [26]	ResNet-50	60.0M	198.5G	30.1	57.2	28.7	31.0	58.7	29.4
	MS R-CNN [26]	ResNet-101	79.0M	251.1G	33.3	61.0	32.9	35.7	63.4	34.7
	Cascade R-CNN [4]	ResNet-50	71.7M	334.1G	25.3	56.1	21.3	29.5	60.8	24.8
	Cascade R-CNN [4]	ResNet-101	90.7M	386.7G	29.5	61.0	25.9	34.6	66.3	31.5
	HTC [7]	ResNet-50	76.9M	331.7G	28.1	56.3	25.1	29.8	59.0	26.6
	HTC [7]	ResNet-101	95.9M	384.3G	30.9	61.0	28.7	34.2	64.5	31.6
	BlendMask [6]	ResNet-50	35.8M	233.8G	28.2	56.4	25.2	27.7M	56.7	24.2
	BlendMask [6]	ResNet-101	54.7M	302.8G	31.2	60.0	28.9	31.4	61.2	28.8
	Mask Transfiner [29]	ResNet-50	44.3M	**185.1**G	28.7	56.3	26.4	29.4	56.7	27.2
	Mask Transfiner [29]	ResNet-101	63.3M	253.7G	31.2	60.7	29.8	34.0	63.1	32.6
One-Stage	YOLACT [3]	ResNet-50	-	-	24.3	53.3	19.7	32.1	65.3	27.9
	YOLACT [3]	ResNet-101	-	-	29.0	60.1	25.3	37.8	70.6	35.6
	CondInst [49]	ResNet-50	**34.1**M	200.1G	30.6	63.6	26.1	33.4	67.4	29.4
	CondInst [49]	ResNet-101	53.1M	269.1G	34.3	67.9	31.6	38.0	71.1	35.6
	QueryInst [19]	ResNet-50	-	-	28.5	60.1	23.1	33.0	66.7	29.4
	QueryInst [19]	ResNet-101	-	-	32.5	65.1	28.6	38.7	72.1	37.6
	SOTR [22]	ResNet-50	63.1M	476.7G	27.9	58.7	24.1	29.3	61.0	25.6
	SOTR [22]	ResNet-101	82.1M	549.6G	32.0	63.6	29.2	34.3	65.7	32.4
	SOLOv2 [57]	ResNet-50	46.2M	318.7G	32.5	63.2	29.9	34.4	65.9	31.9
	SOLOv2 [57]	ResNet-101	65.1M	394.6G	35.2	65.7	33.4	37.8	69.2	36.1
	OSFormer (Ours)	ResNet-50	46.6M	324.7G	41.0	71.1	40.8	42.5	72.5	42.3
	OSFormer (Ours)	ResNet-101	65.5M	398.2G	**42.0**	**71.3**	**42.8**	**44.4**	**73.7**	**45.1**

Real-Time Settings. To improve the application value of OSFormer, we provide a real-time version named OSFormer-550. Concretely, we resize the input shorter side to 550 while reducing the LST encoder layers to 3. As shown in Table. 6, despite the value of AP dropping to 36%, the inference time is increased to $25.8fps$, and the number of parameters and flops is also significantly improved.

Contributions of Different Components. We conduct extensive ablation studies on the COD10K test set [16], including LST encoder (Encoder), location-guided queries (LGQ), blend-convolution FFN (BC-FFN), coarse-to-fine fusion (CFF) and reverse edge attention (REA). We adopt a control variable manner, where we only ablate the current module while keeping the other parts as default settings. When validating the LGQ, we use the learnable embeddings [65] as an alternative. Similarly, BC-FFN is replaced by the vanilla FFN [52]. For the CFF module, we directly use a single-scale feature $T2$ as the output of CFF. As shown in Table. 7, without Encoder, the value of AP directly dropped by about 7%. It indicates that the LST encoder is essential to extract high-level global features. Furthermore, the 2^{nd} row validates the effectiveness of LGQ design again. Note that BC-FFN plays a vital role in the encoder and decoder of LST because

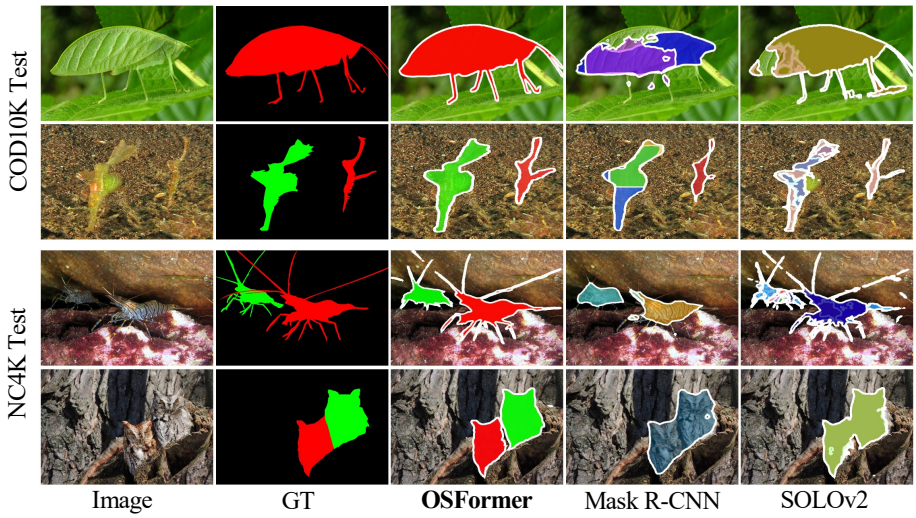


Fig. 7. Qualitative comparison of OSFormer with Mask R-CNN and SOLOv2.

3×3 convolution layers can strengthen the local correlation of global features from self-attention. Moreover, the CFF efficiently fuses multi-scale features and enhances the edges of camouflaged instances by embedding REA. By integrating all modules, OSFormer reaches best performance.

3.2 Comparisons with Cutting-Edge Methods

We compare our OSFormer with several famous instance segmentation models (*i.e.*, two-stage and one-stage models) retrained on the instance-level COD10K [16] dataset. We uniformly adopt official codes for fair comparisons to train each model and evaluate them on the COD10K and NC4K [37] test set. In addition, we also show the results based on different backbones *i.e.*, ResNet-50, ResNet-101, with the ImageNet [14] pre-trained weights.

Quantitative Comparisons. As shown in Table. 8, although the CIS task is challenging, our OSFormer still performs favorably against other competitors across all metrics. In particular, the AP score of OSFormer is higher than that of the second-ranked SOLOv2 [57] by a large margin (∼8.5%) using ResNet-50. The desirable result should be attributed to our LST because it provided higher-level global features and interacted with camouflage clues in different locations in the LST decoder. By leveraging a more powerful backbone, *i.e.*, Swin-T, OSFormer can continue to boost the performance to 47.7% AP (Table. 5). According to the parameters and FLOPs in Table. 8, it also demonstrates that our OSFormer achieves better performance without adding extra parameters.

Qualitative Comparisons. To validate the effectiveness of OSFormer, we also exhibit several representative visualization results in Fig. 7. Specifically, the first

and third row of Fig. 7 shows that our method excels at capturing slender boundaries, which can be attributed to the enhancement of edge features by our REA module. The result of the 2^{nd} row suggests that OSFormer can comfortably distinguish camouflages in the case of multiple instances. Moreover, as illustrated in the last row, OSFormer can still split camouflaged instances even in a coordinated camouflage situation. This is not only due to the global information provided by our LST encoder but also thanks to the sensitivity of our location-guided queries to camouflaged cues in different locations. Overall, compared to the visualization results from other famous methods, OSFormer has the capability to overcome more challenging cases and achieve good performance.

4 Conclusion

We contributed a novel location-aware one-stage transformer framework, called **OSFormer**, for camouflaged instance segmentation (CIS). OSFormer embraces an efficient location-sensing transformer to capture global features and dynamically regress the location and body of the camouflaged instances. As the first one-stage bottom-up CIS framework, we further designed a coarse-to-fine fusion to integrate multi-scale features and highlight the edge of camouflages to produce the global features. Extensive experimental results show that OSFormer performs favorably against all other well-known models. Furthermore, OSFormer requires only about 3,000 images to train, and it converges rapidly. It is easily and flexibly extended to other downstream vision tasks with smaller training samples. We hope this work could provide more basic insights for other potential tasks such as multi-modal camouflaged instance segmentation.

References

1. Bai, M., Urtasun, R.: Deep watershed transform for instance segmentation. In: IEEE CVPR (2017)
2. Bhajantri, N.U., Nagabhushan, P.: Camouflage defect identification: a novel approach. In: IEEE ICIT (2006)
3. Bolya, D., Zhou, C., Xiao, F., Lee, Y.J.: Yolact: Real-time instance segmentation. In: IEEE CVPR (2019)
4. Cai, Z., Vasconcelos, N.: Cascade r-cnn: high quality object detection and instance segmentation. IEEE TPAMI **43**(5), 1483–1498 (2019)
5. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: ECCV (2020)
6. Chen, H., Sun, K., Tian, Z., Shen, C., Huang, Y., Yan, Y.: Blendmask: Top-down meets bottom-up for instance segmentation. In: IEEE CVPR (2020)
7. Chen, K., Pang, J., Wang, J., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Shi, J., Ouyang, W., et al.: Hybrid task cascade for instance segmentation. In: IEEE CVPR (2019)
8. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE TPAMI **40**(4), 834–848 (2017)
9. Chen, S., Tan, X., Wang, B., Hu, X.: Reverse attention for salient object detection. In: ECCV (2018)
10. Chu, H.K., Hsu, W.H., Mitra, N.J., Cohen-Or, D., Wong, T.T., Lee, T.Y.: Camouflage images. ACM TOG **29**(4), 51–1 (2010)
11. Cuthill, I.: Camouflage. JOZ **308**(2), 75–92 (2019)
12. Dai, X., Chen, Y., Yang, J., Zhang, P., Yuan, L., Zhang, L.: Dynamic detr: End-to-end object detection with dynamic attention. In: IEEE CVPR (2021)
13. Dai, Z., Cai, B., Lin, Y., Chen, J.: Up-detr: Unsupervised pre-training for object detection with transformers. In: IEEE CVPR (2021)
14. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: IEEE CVPR (2009)
15. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR (2021)
16. Fan, D.P., Ji, G.P., Sun, G., Cheng, M.M., Shen, J., Shao, L.: Camouflaged object detection. In: IEEE CVPR (2020)
17. Fan, D.P., Ji, G.P., Zhou, T., Chen, G., Fu, H., Shen, J., Shao, L.: Pranet: Parallel reverse attention network for polyp segmentation. In: MICCAI (2020)
18. Fan, D.P., Zhou, T., Ji, G.P., Zhou, Y., Chen, G., Fu, H., Shen, J., Shao, L.: Inf-net: Automatic covid-19 lung infection segmentation from ct images. IEEE TMI **39**(8), 2626–2637 (2020)
19. Fang, Y., Yang, S., Wang, X., Li, Y., Fang, C., Shan, Y., Feng, B., Liu, W.: Instances as queries. In: IEEE CVPR (2021)
20. Fennell, J.G., Talas, L., Baddeley, R.J., Cuthill, I.C., Scott-Samuel, N.E.: The camouflage machine: Optimizing protective coloration using deep learning with genetic algorithms. Evolution **75**(3), 614–624 (2021)
21. Gao, N., Shan, Y., Wang, Y., Zhao, X., Yu, Y., Yang, M., Huang, K.: Ssap: Single-shot instance segmentation with affinity pyramid. In: IEEE CVPR (2019)
22. Guo, R., Niu, D., Qu, L., Li, Z.: Sotr: Segmenting objects with transformers. In: IEEE ICCV (2021)

23. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: IEEE ICCV (2017)
24. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE CVPR (2016)
25. Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance normalization. In: IEEE ICCV (2017)
26. Huang, Z., Huang, L., Gong, Y., Huang, C., Wang, X.: Mask scoring r-cnn. In: IEEE CVPR (2019)
27. Huerta, I., Rowe, D., Mozerov, M., Gonzàlez, J.: Improving background subtraction based on a casuistry of colour-motion segmentation problems. In: Iberian PRIA (2007)
28. Ji, G.P., Chou, Y.C., Fan, D.P., Chen, G., Fu, H., Jha, D., Shao, L.: Progressively normalized self-attention network for video polyp segmentation. In: MICCAI (2021)
29. Ke, L., Danelljan, M., Li, X., Tai, Y.W., Tang, C.K., Yu, F.: Mask transfiner for high-quality instance segmentation. In: IEEE CVPR (2022)
30. Le, T.N., Cao, Y., Nguyen, T.C., Le, M.Q., Nguyen, K.D., Do, T.T., Tran, M.T., Nguyen, T.V.: Camouflaged instance segmentation in-the-wild: Dataset, method, and benchmark suite. IEEE TIP **31**, 287–300 (2022)
31. Le, T.N., Nguyen, T.V., Nie, Z., Tran, M.T., Sugimoto, A.: Anabranch network for camouflaged object segmentation. CVIU **184**, 45–56 (2019)
32. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: IEEE ICCV (2017)
33. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV (2014)
34. Liu, S., Jia, J., Fidler, S., Urtasun, R.: Sgn: Sequential grouping networks for instance segmentation. In: IEEE ICCV (2017)
35. Liu, S., Qi, L., Qin, H., Shi, J., Jia, J.: Path aggregation network for instance segmentation. In: IEEE CVPR (2018)
36. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: IEEE CVPR (2021)
37. Lyu, Y., Zhang, J., Dai, Y., Li, A., Liu, B., Barnes, N., Fan, D.P.: Simultaneously localize, segment and rank the camouflaged objects. In: IEEE CVPR (2021)
38. Matthews, O., Liggins, E., Volonakis, T., Scott-Samuel, N., Baddeley, R., Cuthill, I.: Human visual search performance for camouflaged targets. Journal of Vision **15**(12), 1164–1164 (2015)
39. Mei, H., Ji, G.P., Wei, Z., Yang, X., Wei, X., Fan, D.P.: Camouflaged object segmentation with distraction mining. In: IEEE CVPR (2021)
40. Milletari, F., Navab, N., Ahmadi, S.A.: V-net: Fully convolutional neural networks for volumetric medical image segmentation. In: IEEE 3DV (2016)
41. Mondal, A.: Camouflaged object detection and tracking: A survey. IJIG **20**(04), 2050028 (2020)
42. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: IEEE CVPR (2016)
43. Ren, J., Hu, X., Zhu, L., Xu, X., Xu, Y., Wang, W., Deng, Z., Heng, P.A.: Deep texture-aware features for camouflaged object detection. IEEE TCSVT (2021)
44. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: NeurIPS (2015)
45. Sandon, P.A.: Simulating visual attention. Journal of Cognitive Neuroscience **2**(3), 213–231 (1990)

46. Sofiiuk, K., Barinova, O., Konushin, A.: Adaptis: Adaptive instance selection network. In: IEEE CVPR (2019)
47. Song, L., Geng, W.: A new camouflage texture evaluation method based on wssim and nature image features. In: ICMT (2010)
48. Stevens, M., Merilaita, S.: Animal camouflage: current issues and new perspectives. PTRS B: BS **364**(1516), 423–427 (2009)
49. Tian, Z., Shen, C., Chen, H.: Conditional convolutions for instance segmentation. In: ECCV (2020)
50. Tian, Z., Shen, C., Chen, H., He, T.: Fcos: Fully convolutional one-stage object detection. In: IEEE ICCV (2019)
51. Troscianko, J., Nokelainen, O., Skelhorn, J., Stevens, M.: Variable crab camouflage patterns defeat search image formation. Communications biology **4**(1), 1–9 (2021)
52. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017)
53. Wang, H., Zhu, Y., Adam, H., Yuille, A., Chen, L.C.: Max-deeplab: End-to-end panoptic segmentation with mask transformers. In: IEEE CVPR (2021)
54. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In: IEEE CVPR (2021)
55. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pvtv2: Improved baselines with pyramid vision transformer. CVMJ (2022)
56. Wang, X., Kong, T., Shen, C., Jiang, Y., Li, L.: Solo: Segmenting objects by locations. In: ECCV (2020)
57. Wang, X., Zhang, R., Kong, T., Li, L., Shen, C.: Solov2: Dynamic and fast instance segmentation. In: NeurIPS (2020)
58. Woo, S., Park, J., Lee, J.Y., Kweon, I.S.: Cbam: Convolutional block attention module. In: ECCV (2018)
59. Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., Zhang, L.: Cvt: Introducing convolutions to vision transformers. In: IEEE CVPR (2021)
60. Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J.M., Luo, P.: Segformer: Simple and efficient design for semantic segmentation with transformers. In: NeurIPS (2021)
61. Yan, J., Le, T.N., Nguyen, K.D., Tran, M.T., Do, T.T., Nguyen, T.V.: Mirrornet: Bio-inspired camouflaged object segmentation. IEEE Access **9**, 43290–43300 (2021)
62. Yang, F., Zhai, Q., Li, X., Huang, R., Luo, A., Cheng, H., Fan, D.P.: Uncertainty-guided transformer reasoning for camouflaged object detection. In: IEEE CVPR (2021)
63. Zhai, Q., Li, X., Yang, F., Chen, C., Cheng, H., Fan, D.P.: Mutual graph learning for camouflaged object detection. In: IEEE CVPR (2021)
64. Zhu, J., Zhang, X., Zhang, S., Liu, J.: Inferring camouflaged objects by texture-aware interactive guidance network. In: AAAI (2021)
65. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. In: ICLR (2020)