Inferring Camouflaged Objects by Texture-Aware Interactive Guidance Network

Jinchao Zhu, [1,2] **Xiaoyu Zhang,** [1,2*] **Shuo Zhang,** [1,2] **Junnan Liu** [3]
[1] Institute of Robotics and Automatic Information System, College of Artificial Intelligence, Nankai University, Tianjin, China
[2] Tianjin Key Laboratory of Intelligent Robotics, Nankai University, Tianjin, China
[3] College of Intelligent Systems Science and Engineering, Harbin Engineering University, Harbin, China
jczhu@mail.nankai.edu.cn, zhangxiaoyu@nankai.edu.cn, 1711507@mail.nankai.edu.cn, 1070715836@hrbeu.edu.cn

Abstract

Camouflaged objects, similar to the background, show indefinable boundaries and deceptive textures, which increases the difficulty of detection task and makes the model rely on features with more information. Herein, we design a texture label to facilitate our network for accurate camouflaged object segmentation. Motivated by the complementary relationship between texture labels and camouflaged object labels, we propose an interactive guidance framework named TINet, which focuses on finding the indefinable boundary and the texture difference by progressive interactive guidance. It maximizes the guidance effect of refined multi-level texture cues on segmentation. Specifically, texture perception decoder (TPD) makes a comprehensive analysis of texture information in multiple scales. Feature interaction guidance decoder (FGD) interactively refines multi-level features of camouflaged object detection and texture detection level by level. Holistic perception decoder (HPD) enhances FGD results by multi-level holistic perception. In addition, we propose a boundary weight map to help the loss function pay more attention to the object boundary. Sufficient experiments conducted on COD and SOD datasets demonstrate that the proposed method performs favorably against 23 state-of-the-art methods.

Introduction

Salient object detection (SOD) (Borji et al. 2019) aims to estimate the visual salient regions, while camouflaged object detection (COD) aims to identify the concealed objects that similar to their surroundings and the camouflaged objects usually are not easy to find. So COD is more prone to serious misjudgment than SOD. They both can be used as initial steps of many image comprehension and processing tasks, such as image editing, image retrieval, photo composition, and target tracking. Besides, COD research will contribute to the development of medical image, military, agricultural, and wildlife protection fields. Examples include pneumonia segmentation, polyp segmentation (Fan et al. 2020b), tumor segmentation, camouflage enemy facility detection, detection of burial mines, locust detection and rare animal detection.

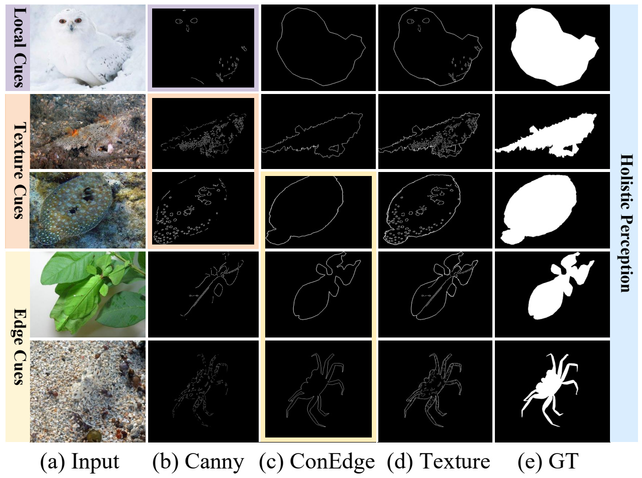

Figure 1: Given scenes like (a), how to find the camouflage objects? We propose TINet to infer the objects from local saliency cues, texture cues (b), contour edge cues (c), and holistic perception (e), which is consistent with the process of human inferring and thinking. ConEdge is object contour extracted from GT. (b) is the Canny map of (a) and it only shows the object areas. (d) is (b) + (c).

A sufficiently large and finely marked COD dataset was proposed by (Fan et al. 2020a). Most of the camouflaged objects in the dataset are animals and they usually adapt the color of their body to match the surroundings or hide in an environment close to their color to avoid recognition (Owens et al. 2014). The camouflage strategy works by deceiving and misleading the visual perceptual system of the observer. We can feel the effect of the strategy by trying to find the camouflaged objects and identify the full boundaries of the camouflaged objects in Fig. 1.

Salient objects have obvious stimulation to the visual perceptual system of the observer and can be identified instantaneously. However, the process of camouflage object detection is a longer thinking process of anti-deception, where more types of cues are used. For example, for the first row of Fig. 1, we first find the owl's face (local saliency) and then follow this clue to confirm the overall outline. For the 2,3 rows, when the colors of the camouflaged objects are sim-

*Corresponding author

ilar to the environment and the edges are blurred, the texture features (texture cues) will be an important clue. For the 4,5 rows, the texture of the targets is deceptive and almost identical to the background, but relatively clear edges (edge cues) and unique shapes help to identify the targets. In addition, holistic perception plays an important role. Inspired by the above ideas, we improved the contour edge label (Fig. 1(c) ConEdge) to obtain the texture label (Fig. 1(d) Texture) by adding detailed information about the surface of objects, which is helpful to discover the difference between the texture of the camouflaged objects and the background. The texture label here doesn't just represent texture, it combines multiple cues: local saliency cues, texture cues, and edge cues.

In this paper, our proposed texture-aware interactive guidance network (TINet) focuses on the progressive interactive guidance of multi-level texture features and multi-level segmentation features. Specifically, TINet has three decoders: texture perception decoder (TPD), feature interaction guidance decoder (FGD), and holistic perception decoder (HPD). TINet is a dual-task model. HPD and TPD complement each other in their respective tasks. Taking the segmentation task as an example, TPD makes a comprehensive analysis of texture information. FGD interactively refines multi-level features of camouflaged object detection and texture detection progressively from bottom to top level by level. FGD-T and FGD-S are two components of FGD coupled with each other. In FGD-T, two types of high-level features with rich semantic information and accurate location information can effectively suppress the background noise of the low-level texture features. In FGD-S, the optimized texture features contain sufficient internal details, which helps to infer more accurate segmentation features. HPD enhances FGD results through feedback and integration. In addition, we use weight maps to optimize the loss function. In order to verify the universality of the proposed method, we train two models respectively for COD and SOD tasks. According to the characteristics of SOD task, we adjust the structure of the network slightly and use contour edge labels to get the best results.

To sum up, our contributions are as follows:

- We propose a texture label with multiple cues and design FGM modules for heterogeneous feature fusion which use two types of high-level features with rich semantic information to guide one type of low-level features with rich details.
- We introduce the FGD decoder for bidirectional feature optimization, where segmentation features suppress the background noise of texture features and the refined texture features are used to infer more accurate segmentation features. TPD and HPD decoders further optimize the results of FGD from the perspective of texture perception and holistic perception.
- Sufficient experiments conducted on 4 COD and 5 SOD datasets demonstrate that the proposed method outperforms 13 state-of-the-art COD methods and 23 state-of-the-art SOD methods in terms of eight metrics, which proves the advancement of our method.

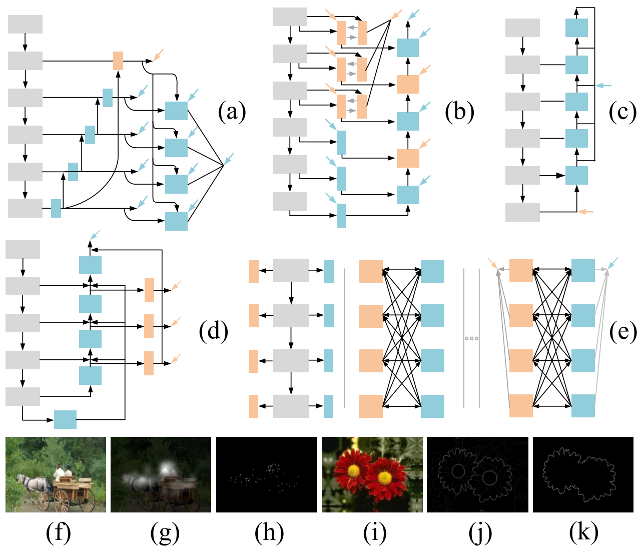

Figure 2: Simplified structure of multi-cue models. Gray, blue and orange blocks are encoders, decoders and edge modules. The blue and orange arrows are mask labels and other types of labels, respectively. (a) EGNet (b) MLMSNet (c) ASNet (d) PoolNet (e) SCRN (j) StdEdge (k) ConEdge (g) and (h) are two types of eye fixation maps.

Related Work

Limited by the lack of rich datasets, the research in the COD field is still in its infancy. The author of COD10K dataset (Fan et al. 2020a) proposed SM and IM modules to realize search and recognition, which are inspired by hunting. Because generic object detection, salient object detection, and camouflaged object detection are three related detection tasks (Zhao et al. 2019b), we can find the inspiration for COD from the excellent algorithms of SOD, which has been verified in the experiment of (Fan et al. 2020a). Early traditional SOD approaches mainly rely on hand-crafted features (Yang et al. 2013; Zhu et al. 2014; Zhang et al. 2015) and a comprehensive survey (Borji et al. 2015) shows it. With the development of CNNs, scholars have proposed various methods to aggregate multi-level features to produce excellent results.

Deep Aggregation Models

(Long, Shelhamer, and Darrell 2015) proposed a creative FCN network to predict images at the pixel level. (Hou et al. 2017) added shortcut connections to integrate features at different levels within the HED architecture (Xie and Tu 2015). (Chen et al. 2020) designed the modules FIA, HA, SR, and GCF to make up the gap between different features and solved the dilution problem in the feature transfer process. (Feng, Lu, and Ding 2019) proposed the attentive feedback modules and boundary-enhanced loss to optimize the structure and boundaries of salient objects. (Wang et al. 2019c) designed an essential pyramid attention structure, which enables the network to concentrate more on salient regions. (Wang et al. 2019a; Wei, Wang, and Huang 2020; Wu, Su, and Huang 2019b) all adapted the recurrent structure to iteratively perform refinement. (Pang et al. 2020) proposed AIM and SIM modules to better detect size-varying objects and summarized the structure of multiple deep aggregation models by simplified diagrams. We also show the simplified structure of multi-cue models in Fig. 2.

Multi-Cue Models

Contour edge cue: (Zhao et al. 2019a) explicitly modeled edge information for salient objects to solve the problem of coarse object boundaries. (Wu, Su, and Huang 2019b) designed a stacking cross refinement unit to simultaneously refine multi-level features of salient object detection and edge detection. In SOD task, different from EGNet, our structure is multi-level interactive guidance and the internal and background noises of edge cues are suppressed by segmentation features. Compared with S$\rightarrow$E and E$\rightarrow$S structure of SCRN, TINet has a sparse guiding structure and uses two types of features to guide one type of feature: E,S$\rightarrow$S and E,S$\rightarrow$E.

Standard edge cue: (Liu et al. 2019) expanded the role of pooling in CNN by feature aggregation modules and demonstrated that the guidance effect of standard edge (StdEdge) from edge dataset (Arbelaez et al. 2011; Mottaghi et al. 2014) is better than contour edge (ConEdge) through experiments. (Wu et al. 2019) utilized the StdEdge detection and ConEdge detection together to improve the performance. They both showed that the inner and outer edges of the foreground contour are valuable, which indirectly shows that the texture label we proposed that contains the inner details is valuable for camouflaged objects with blurred boundaries.

Eye fixation cue: (Wang et al. 2019b) built an attentive saliency network that learns to detect salient objects from fixation maps that can be seen as the locally salient location of the salient objects. As shown in Fig. 2 (g) (h), eye fixation labels are in the form of spots or clouds similar to the local saliency situation in Fig. 1. Therefore, in the case of blurred foreground contour, we speculated that the internal details would help to locate the camouflaged objects.

Proposed Method

In this section, we first explore the logical interrelations between the contour edge maps, texture maps, and ground truth. Then we discuss the functions of the feature interaction guidance decoder and perception decoders. Finally, we compare two interactive structures of TINet for SOD and COD tasks in detail and introduce the boundary pixel weight map.

Texture Label of TINet

The objects in SOD have clear boundaries and distinctive foreground colors, which can be used as clues to find objects. The objects in COD are not so conspicuous and usually have fuzzy boundaries, camouflage colors close to the background, camouflage shape, or even camouflage texture. By observing the COD dataset, we find that the human thinking process of discovering the objects and confirming their boundaries can be summarized as follows: inferring the whole object by the local saliency region, local clear boundaries, or unique shape; comparing the texture of the object with the background. We use the Canny algorithm to find the obvious lines inside the objects and add contour edges to get the texture map, which can help us find the above cues. When making a texture label, the image is smoothed with 9x9 Gaussian kernel and the Canny threshold is 50-150. We call them texture maps because these lines can represent the texture of the objects abstractly. The relationship between contour edge map (ConEdge), texture map (Texture), and GT is as follows:

$$ConEdge + Canny \times GT = Texture \tag{1}$$

Perception Decoder

In the training stage, the perception decoders (PD) perceive the camouflaged objects from different angles, including holistic perception, edge perception, and texture perception. All decoders participate in the training to help optimize the parameters. In the testing stage, PDs refine the results of the feature interactive guidance decoder (FGD) by means of feedback and integration. Sometimes higher-level features will lose some important feature information during feature extraction. So the output features of the top FGM supervised by interlayer supervision (ILS) are downsampled and fed back to previous layers to refine them. After the top-down feedback process, TINet uses perception modules (PM) to progressively integrate features from down to up. Besides, if only a segmentation task is needed, the PD on the left (Fig. 3 TPD) is not involved in the calculation and vice versa.

The perception module (PM) realizes the fusion of features of the same type. PM can be divided into three types according to the different supervision labels and they are texture perception module (TPM) for COD, edge perception module (EPM) for SOD, and holistic perception module (HPM). Generally, high-level features have richer semantic information, while low-level features preserve more details including background noise and foreground details. In Fig.3, the HPM utilize the high-level segmentation features f_s^i to guide the low-level segmentation features f_s^{i-1} to obtain new low-level segmentation features. The new features will be fed to the next HPM. In detail, HPM has two steps: first, high-level features f_s^i and low-level features f_s^{i-1} are fused by element-wise multiplication, where background noise of f_s^{i-1} is suppressed and foreground details are preserved. Then the fused features are used to guide the low-level features f_s^{i-1} by element-wise addition. The element-wise multiplication is relatively rough. If the semantic information of high-level features is not accurate, some useful details of low-level features will be mistakenly suppressed. So the element-wise addition can also be understood as supplementing the details of low-level features to prevent the wrong fusion of high-level features. There are CBR (convolution, batch normalization, and Relu) before and after each operation. The process of PM can be represented as follows:

$$f_s^{i-1} = C(f_{HPM}^{i-1}) = C(C(f_s^{i-1}) + C(C(f_s^i) * C^2(f_s^{i-1}))) \tag{2}$$

$$f_t^{i-1} = C(f_{TPM}^{i-1}) = C(C(f_t^{i-1}) + C(C(f_t^i) * C^2(f_t^{i-1}))) \tag{3}$$

where each of $C(\cdot)$ is CBR. i represents the level of the features. $C^2(\cdot)$ indicates that the features pass through two CBRs. $*$ is element-wise multiplication. f_{HPM}^{i-1} and f_{TPM}^{i-1} are features before passing through the last CBR in Fig.3 HPM, TPM.

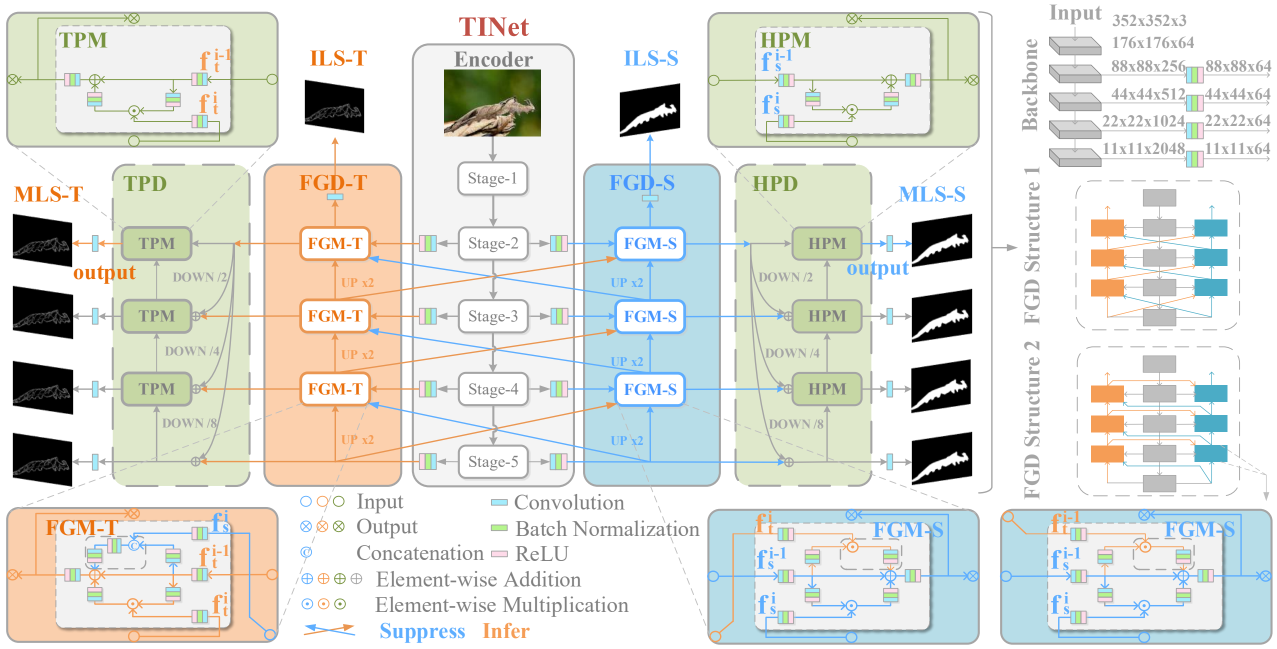

Figure 3: An overview of the proposed TINet. ResNet-50 is used as the backbone encoder. FGD-S and FGD-T realize interactive guidance. PD consists of feedback and integration components. Multiple PMs form the integration part. Multi-level supervision and interlayer supervision (MLS and ILS) are used to assist training. Features outside the encoder are unified into 64 channels.

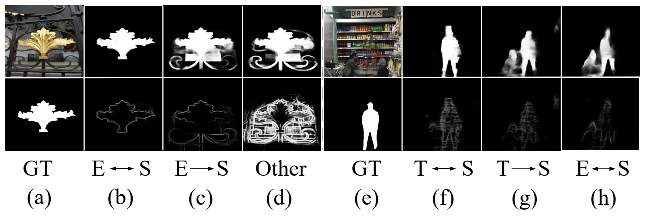

Figure 4: (b) (c) (f) (g) show the effect of interactive guidance. (b) (d) show the effect of multi-level interaction optimization. (d) is the edge map of EGNet generated by explicitly modeling only the lowest level edge features. (f) (h) show the advantages of texture features over edge features.

Feature Interaction Guidance Decoder

In Fig.3, FGD-S and FGD-T are complementary decoders composed of multiple FGM-S and FGM-T modules, which can effectively use multi-level segmentation features and texture features to progressively refine the camouflaged object detection and texture detection level by level. For FGD-S, the high-level segmentation and texture features with rich semantic information help the low-level segmentation features suppress background noise and save the detailed information in camouflaged object areas. The internal high-level texture information of the camouflaged object helps to locate and infer the whole camouflaged object. For FGD-T, two high-level guiding features effectively suppress background noise and constrain the diffusion of low-level texture features. The refined texture features contain sufficient internal details, which helps to infer more accurate segmentation features. Fig. 4 (f) (g) show the difference between with and without guidance S→T.

The feature interaction guidance module (FGM) is an enhanced version of PM. FGM module adds another type of high-level features to achieve dual guidance on the basis of PM. FGM-S adds the guidance of high-level texture features f_t^i to low-level segmentation features f_s^{i-1}. Element-wise multiplication is used to enhance the localization effect of texture features on segmentation features. FGM-T is similar to FGM-S. The difference is that FGM-T use concatenation operation to merge the high-level segmentation features f_s^i with low-level texture features f_t^{i-1} on channel axis for suppressing background noise and preventing the diffusion of inaccurate texture features. Here, another CBR is adopted to turn 128 channels back to 64. The processes in the dotted boxes in Fig. 3 FGM-S, FGM-T can be described as follows:

$$f_{FGM-S}^{i-1} = C(C(f_t^i) * C^2(f_s^{i-1})) \tag{4}$$

$$f_{FGM-T}^{i-1} = C(C(Cat(C(f_s^i), C^2(f_t^{i-1})))) \tag{5}$$

f_{FGM-S}^{i-1} and f_{FGM-T}^{i-1} are the outputs of the dotted boxes. FGM-S and FGM-T both use two types of high-level features to guide and optimize one type of low-level features. The whole process is expressed as follows:

$$f_s^{i-1} = C(f_{HPM}^{i-1} + f_{FGM-S}^{i-1}) \tag{6}$$

$$f_t^{i-1} = C(f_{TPM}^{i-1} + f_{FGM-T}^{i-1}) \tag{7}$$

For f_{HPM}^{i-1} and f_{TPM}^{i-1}, please refer to Eq. 3.

Interactive Guidance Structure

In this section, we compare two interactive structures for SOD and COD tasks. On the right side of Fig.3, we get structure 2 by changing the input f_t^i of the FGM-S module to

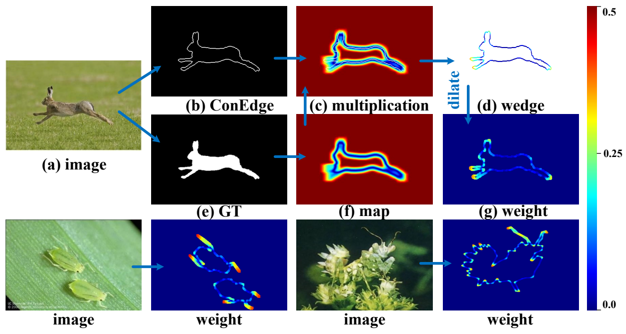

Figure 5: The calculation process of weight map: (b) and (f) are multiplied to get the weight of edge pixels (d). (d) is dilated to obtain (g). The last row shows more examples.

f_t^{i-1} and compare the supervision effects of ConEdge and texture labels. We find that texture label and structure 1 have the best performance in COD task and ConEdge label and structure 2 have the best performance in SOD task, which is shown in Tab. 2. In COD task, the low-level texture features f_t^{i-1} of camouflaged object are not clear and easy to generate error guidance. However, high-level texture features f_t^i are beneficial to localization, so the guidance effect of high-level texture features is better. In SOD task, the boundary of salient object is relatively clear and the low-level edge features have high credibility. Therefore, the guidance effect of relatively accurate low-level edge features f_e^{i-1} is better. Fig. 4 (b) (c) (f) (g) show the effectiveness of interactive guidance. (d) is the edge map of EGNet generated by explicitly modeling and only the lowest level edge features are used. (b) shows the effect of multi-level interactive optimization of our model. (g) (h) show the advantages of texture features over contour edge features in COD.

Boundary Pixel Perception Loss

The accuracy of the boundary directly affects the detection result, so BPP loss with corner perception capability focuses on the boundary and trains the pixels around it differently. Usually, the straight boundaries are relatively easy to detect and the obvious raised and sunken parts are easily misjudged. BPP initially assigns weights to get $map_{i,j}$ according to the GT, as shown in Eq.8. In Eq.9, we get the weight of the ConEdge pixels. Strengthening the training of the pixels on both sides of the boundary helps the model distinguish the local details accurately. Eq.10 uses the dilation operation to spread the influence range of the ConEdge weight. The final weight map (g) is used to improve binary cross entropy (BCE) and IoU loss. IoU loss helps to optimize the global structure (Rahman and Wang 2016). BCE and IOU treat pixels equally. We design (g) w to make up for this weakness and help them focus on pixels near key boundaries.

$$map_{ij} = \left| \frac{\sum_{m,n\epsilon R_{ij}} gt_{mn}}{\sum_{m,n\epsilon R_{ij}} 1} - 0.5 \right| \quad (8)$$

$$wedge_{ij} = ConEdge \times map_{ij} \quad (9)$$

$$w_{ij} = dilate(wedge_{ij}) \quad (10)$$

$$L_{bce}^w = -\frac{\sum_{i=1}^H \sum_{j=1}^W (1+w_{ij}) \sum_{l=0}^1 \mathbf{1}(g_{ij}=l) log \mathbf{Pr}(p_{ij}=l|\psi)}{\sum_{i=1}^H \sum_{j=1}^W (1+w_{ij})} \quad (11)$$

$$L_{iou}^w = 1 - \frac{\sum_{i=1}^H \sum_{j=1}^W (gt_{ij} * p_{ij}) * (1+w_{ij})}{\sum_{i=1}^H \sum_{j=1}^W (gt_{ij} + p_{ij} - gt_{ij} * p_{ij}) * (1+w_{ij})} \quad (12)$$

$$L_w = L_{bce}^w + L_{iou}^w \quad (13)$$

$$L_{bpp} = L^s + L^e = \sum_{i=2}^5 \frac{1}{2^{i-1}} L_w^{si} + \frac{1}{2} L_w^{ils-s} + \sum_{j=2}^5 \frac{1}{2^{j-1}} L_w^{ej} + \frac{1}{2} L_w^{ils-e} \quad (14)$$

Hyperparameter setting: R_{ij} is a square with a side length of 25 centered on (i,j), and (m,n) is a point in R_{ij}. We use max-pooling and avg-pooling to implement dilation, which is explained in detail in the supplementary materials. In Eq.11, $\mathbf{1}(\cdot)$ is the indicator function and $l\epsilon\{0,1\}$ is pixel label. p_{ij} and g_{ij} are the model prediction and ground truth of the pixel at (i,j). $\mathbf{Pr}(\cdot)$ refers to the prediction probability. ψ represents the model parameters. In Eq.14, L_w^{si} and L_w^{ej} are multi-level supervision for optimizing training. L_w^{ils-s} and L_w^{ils-e} are interlayer supervision.

Experiments

Datasets and Evaluation Metrics

We evaluate the proposed architecture on four COD datasets: CAMO-Test (Le et al. 2019), CPD1K-Test (Zheng et al. 2019), CHAMELEON, COD10K-Test (Fan et al. 2020a) and five SOD datasets: ECSSD (Yan et al. 2013), PASCAL-S (Li et al. 2014), DUT-OMRON (Li et al. 2014), DUTS, HKU-IS-Test (Li and Yu 2015). Six metrics are adopted to evaluate the performance of TINet and other models. The first two metrics are mean absolute error (MAE) (Perazzi et al. 2012) and F-measure (F_β) (Achanta et al. 2009), which are widely used in previous models (Fan et al. 2020c; Zhang et al. 2020). F_β is formulated as the weighted harmonic mean of Precision and Recall, β^2 is generally set to 0.3. We calculate the maximal F_β values from PR curves, denoted as F_{max} and use an adaptive threshold that is twice the mean value of the prediction to calculate F_{avg}. Weighted F-measure (F_β^w) defines a weighted precision, which is a measure of completeness for improving F-measure. Structural similarity measure ($S_\alpha, \alpha = 0.5$) (Fan et al. 2017) and E-measure (E_m) (Fan et al. 2018) are also useful for evaluation of saliency maps. For MAE, smaller is better. For other metrics, bigger is better.

Implementation

We combine the training datasets of CAMO-Train, CPD1K-Train, COD10K-Train and take them as the COD training dataset, which follows SINet (Fan et al. 2020a). We use DUTS-TR (Wang et al. 2017) as training dataset for SOD. ResNet-50 (He et al. 2016) are adopted as backbone. Horizontal flip, random crop, and multi-scale input images are used for data augmentation. Warm-up and linear decay strategies are used. The maximum learning rate is 5e-3 for the backbone and 0.05 for other parts. Stochastic gradient descent is adopted to train the network with the momentum

COD																
COD-Model	CAMO-TE				CHAMELEON-TE				COD10K-TE				CPD1K-TE			
	F_β^w	E_m	S_m	MAE	F_β^w	E_m	S_m	MAE	F_β^w	E_m	S_m	MAE	F_β^w	E_m	S_m	MAE
CPD_{19}^R	.550	.802	.726	.115	.706	.878	.853	.052	.508	.763	.747	.059	.589	.700	.828	.010
EGNet_{19}^R	.616	.832	.748	.100	.752	.897	.871	.042	.574	.808	.774	.047	.690	.790	.870	.007
SCRN_{19}^R	.635	.847	.770	.091	.748	.894	**.874**	.043	.588	.799	.790	.046	.652	.723	.862	.010
ITSD_{20}^R	.611	.833	.754	.099	.689	.873	.832	.053	.570	.796	.773	.049	.405	.614	.726	.017
GateNet_{20}^R	.643	**.850**	.768	.091	.756	.897	**.874**	.041	.602	.807	**.793**	.043	690	780	869	.007
GCPANet_{20}^R	.590	.821	.759	.108	.681	.866	.847	.053	.526	.755	.766	.056	.638	.722	.857	.008
MINet_{20}^R	.629	.832	.746	.093	.771	**.929**	.855	**.036**	.607	.848	.770	**.042**	.706	.876	.847	.006
SINet_{20}^R	.606	.834	.751	.100	.740	.899	.869	.044	.551	.797	.771	.051	.587	.725	.849	.010
F3Net_{20}^R	.664	.839	.779	.091	.765	.909	.867	.041	.615	.836	.787	.046	.754	.916	.873	.006
Ours(E+STR1)	.668	.843	.774	.089	.774	.904	.866	.041	.625	.833	.789	.044	.748	.897	.874	.006
Ours(T+STR1)	**.678**	.847	**.781**	**.087**	**.783**	.916	**.874**	.038	**.635**	**.848**	**.793**	.043	**.787**	**.930**	**.885**	**.005**

SOD																				
SOD-Model	DUTS-TE				DUT-OMRON				HKU-IS				ECSSD				PASCAL-S			
	F_β^w	E_m	S_m	MAE	F_β^w	E_m	S_m	MAE	F_β^w	E_m	S_m	MAE	F_β^w	E_m	S_m	MAE	F_β^w	E_m	S_m	MAE
CPD_{19}^R	.795	.904	.869	.043	.719	.873	.825	.056	.876	.952	.906	.034	.898	.949	.918	.037	.803	.887	.847	.072
EGNet_{19}^R	.816	.907	.887	.039	.738	.874	.841	.053	.887	.956	.918	.031	.903	.947	.925	.037	.807	.879	.853	.075
PoolNet_{19}^R	.817	.912	.887	.037	.725	.874	.831	.054	.888	.959	.919	.030	.904	.948	.926	.035	.823	.890	**.867**	.065
SCRN_{19}^R	.803	.902	.885	.040	.720	.869	.837	.056	.878	.954	.917	.033	.900	.942	**.927**	.037	.816	.888	**.867**	.065
GCPANet_{20}^R	.821	.913	**.891**	.038	.735	.869	.839	.056	.889	.956	.920	.031	.900	.950	.925	.036	.821	**.901**	.866	**.062**
MINet_{20}^R	.825	.917	.884	.037	.738	.873	.833	.055	.899	**.961**	.920	.028	.911	**.953**	.925	**.033**	.821	.899	.857	.064
F3Net_{20}^R	.835	.918	.888	**.035**	.747	**.876**	.839	.053	.900	.959	.917	.028	.912	.946	.924	**.033**	.827	.895	.860	.063
Ours(E+STR1)	.836	.918	**.891**	**.035**	.748	**.876**	.840	.053	.905	.960	.921	**.027**	.911	.950	.924	.034	.826	.897	.861	**.062**
Ours(E+STR2)	**.842**	**.925**	**.891**	**.035**	**.754**	**.876**	**.842**	**.051**	**.906**	.960	**.922**	**.027**	**.914**	**.953**	.926	**.033**	**.829**	**.901**	.861	**.062**

Table 1: Quantitative evaluation for COD and SOD. We compare 13 COD and 23 SOD methods on 4 COD and 5 SOD datasets. E, T, STR1, STR2 stand for ConEdge label, Texture label, and Structure 1, 2. R: ResNet-50, ResNeXt-101 or ResNet-101 as backbone; V: VGG16 as backbone. The comparison algorithms before 2019 are introduced in the experimental analysis section.

of 0.9 and the weight decay of 5e-4. Batchsize and maximum epoch are set to 32 and 45 respectively. We train the model on a PC with 16GB RAM and an RTX 2080Ti GPU. We resize images to 352×352 in the inference stage.

Comparative Experiment of Label and Structure

For COD and SOD, we use different labels and structures to compare the effects of different combinations in Tab.2. The boundaries of the objects in the COD dataset are fuzzy, so the high-level texture features refined by segmentation features have better inference and guidance effect on the low-level segmentation features. However, the boundaries of the objects in the SOD task are relatively clear, the low-level edge features optimized by high-level segmentation features can guide the same level segmentation features well. ConEdge+Stucture2 and Texture+Struture1 are the best combinations for SOD and COD, respectively. We use these combinations in ablation experiments. It is worth noting that the loss of the above experiments is optimized by weight map. We use the weight map of (Wei, Wang, and Huang 2020) to perform the experiment of the COD part in Tab.2 again. The performance of the texture label in structure 1 is better than the ConEdge label.

Ablation Analysis

To analyze the role of various components of TINet, a series of experiments are shown in Tab.3. R, G, and P represent backbone (ResNet-50), interactive guidance decoder (FGD), and perception decoder (PD). RGP is supervised by segmentation labels and interactive guidance doesn't work. After adding the guidance of texture (edge) features to segmentation features, the performance is improved. At this time, because the texture (edge) features are not refined by the high-level segmentation features, the guidance effect can be further improved. In bi-directional guidance ($\leftrightarrow$), the segmentation features suppress the outward diffusion of texture features and the optimized texture features can effectively infer the segmentation features. The same is true for edge features in the SOD task.

Multi-level supervision, BCE loss, and IoU loss are used in the experiments. Because these training methods have been verified in many algorithms, we only add weighted loss in the last 4 experiments to verify the effect of weight maps. In the 6th line, we compare the scores of output in ILS supervision position to illustrate the improvement brought by HPD. The 7th line shows the difference between concatenation and multiplication operations in FGD-T(E). When concatenation is replaced by multiplication, the scores will get worse. The 8th line removes the guidance of the same type features in FGM-T(E) and FGM-S to verify the effect of dual guidance. Baseline is the RGP without IoU loss.

Comparison with State-of-the-arts

We compare the proposed method with 13 state-of-the-art methods for COD, including CPD (Wu, Su, and Huang 2019a), EGNet, SCRN (Wu, Su, and Huang 2019b), ITS-

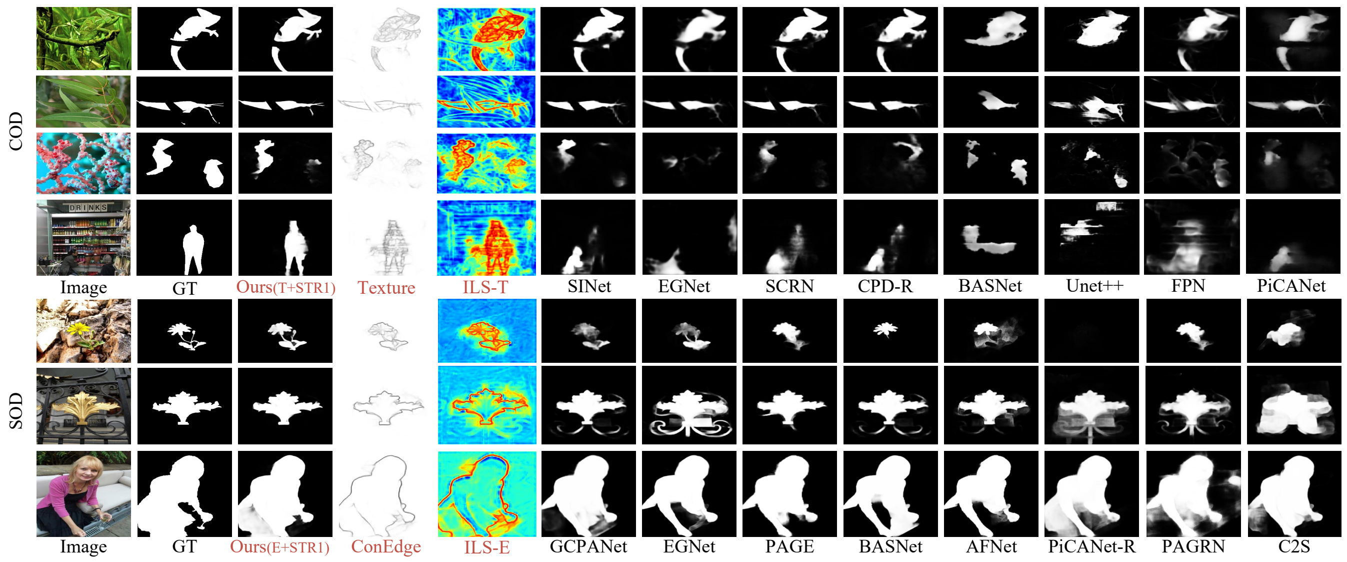

Figure 6: Qualitative comparisons with state-of-the-art algorithms.

Model	COD				SOD			
	CAMO		CHAME		OMRON		PASCAL-S	
	F_β^w	MAE	F_β^w	MAE	F_β^w	MAE	F_β^w	MAE
E+S2	.674	.089	.770	.040	.754	.051	.829	.062
T+S2	.668	.091	.773	.039	.747	.055	.825	.064
E+S1	.668	.089	.774	.041	.748	.053	.826	.062
T+S1	.678	.087	.783	.038	.734	.056	.824	.064

Table 2: Comparative experiment of label and structure. E, T, S1, S2 stand for ConEdge, Texture labels, Structure 1, 2.

D (Zhou et al. 2020), GateNet (Zhao et al. 2020), GCPANet (Chen et al. 2020), MINet, SINet, F3Net (Wei, Wang, and Huang 2020), etc. 23 SOD methods used for comparison are CPD, EGNet, PoolNet (Liu et al. 2019), SCRN, GCPANet, MINet. F3Net, etc. For fair comparisons, we use saliency maps provided by the authors or generated by their codes. To save space, Tab.1 does not show the algorithms of earlier years, which are the comparative algorithms in the papers of SINet and MINet (Pang et al. 2020).

Quantitative Evaluation Tab. 1 COD part shows the F_β^w, E_m, S_m, MAE scores of the proposed model and COD state-of-the-arts algorithms on 4 COD datasets recommended by (Fan et al. 2020a). Due to the positioning and inferring function of texture features, texture label brings a significant improvement to the network compared with ConEdge label under the same network structure 1. Tab. 1 SOD part shows the scores of the proposed model and SOD state-of-the-art algorithms on 5 widely used datasets and proves that the guidance effect of low-level edge features is better. It can be seen that our model performs favorably against other methods in terms of multiple measures on two types of tasks, which demonstrates the effectiveness of our network.

Qualitative Evaluation Fig. 6 shows the visual examples produced by our model and others. We visualize the features in the ILS-T position to show how our model enhances the internal texture of the objects and suppresses the background. The refinement of texture features is synchronized with the optimization of segmentation features. Our method performs better in various challenging scenarios.

Model	COD (T+S1)		SOD (E+S2)	
	CHAME		DUTS-TE	
	F_β^w	MAE	F_β^w	MAE
Baseline	.686	.053	.808	.042
RGP	.760	.044	.828	.040
PGRGP[T(E)$\to$S]	.764	.042	.830	.039
PGRGP[T(E)$\leftrightarrow$S]	.772	.041	.834	.037
PGRGP[T(E)$\leftrightarrow$S] + w	.783	.038	.842	.035
ILS	.775	.039	.839	.036
w/o Cat	.773	.040	.837	.037
w/o T(E)$\to$T(E), S$\to$S	.767	.042	.834	.038

Table 3: Ablation analysis. R, G, and P represent backbone (ResNet-50), interactive guidance decoder (FGD), and perception decoder (PD), respectively. $\to$ represents single direction guidance of texture (or ConEdge) features to segmentation features. $\leftrightarrow$ represents bi-directional guidance. w indicates that the loss function uses the weight map.

Conclusions

In this paper, we propose a structurally symmetric framework named TINet, which interactively refines multi-level texture and segmentation features by FGD decoders. HPD and TPD (EPD) decoders realize the integration function to improve the results of FGD. The edge weight map is designed to help the model pay more attention to the pixels around the key parts of the boundaries. By slightly adjusting the structure and label, our models perform favorably against the state-of-the-art models on COD and SOD datasets.

Acknowledgements

This work was supported by Nankai University. We would like to thank reviewers for their constructive suggestions.

Appendix

Parameter Setting

The dilation operation in Eq.10 is implemented by max-pooling and avg-pooling. Because the variable parameters include the size and type of pooling, to simplify the experiment, we fixed the size to adjust the pooling type. The sizes of the three pooling operations are fixed to 3x3, 11x11, and 21x21, respectively. Since the max-pooling and the avg-pooling have different effects, we adjust their combinations to achieve the best results. The dilation operation helps to spread the influence range of the ConEdge weights. The closer to the edge, the greater the weight of the pixels. The optimal weight map is obtained by the following equation:

$$\begin{aligned} w_{ij} &= wedge_{ij} + maxpooling_{3\times3}(wedge_{ij}) \\ &+ avgpooling_{11\times11}(wedge_{ij}) + avgpooling_{21\times21}(wedge_{ij}) \end{aligned} \quad (15)$$

References

Achanta, R.; Hemami, S.; Estrada, F.; and Susstrunk, S. 2009. Frequency-tuned salient region detection. In *CVPR*, 15971604.

Arbelaez, P.; Maire, M.; Fowlkes, C. C.; and Malik, J. 2011. Contour Detection and Hierarchical Image Segmentation. *IEEE TPAMI* 33(5): 898–916.

Borji, A.; Cheng, M.; Hou, Q.; Jiang, H.; and Li, J. 2019. Salient object detection: A survey. *Computational Visual Media* 5(2): 117–150.

Borji, A.; Cheng, M.; Jiang, H.; and Li, J. 2015. Salient Object Detection: A Benchmark. *IEEE TIP* 24(12): 5706–5722.

Chen, Z.; Xu, Q.; Cong, R.; and Huang, Q. 2020. Global Context-Aware Progressive Aggregation Network for Salient Object Detection. In *AAAI*.

Fan, D.; Cheng, M.; Liu, Y.; Li, T.; and Borji, A. 2017. Structure-Measure: A New Way to Evaluate Foreground Maps. In *ICCV*, 4558–4567.

Fan, D.; Gong, C.; Cao, Y.; Ren, B.; Cheng, M.; and Borji, A. 2018. Enhanced-alignment Measure for Binary Foreground Map Evaluation. In *IJCAI*, 698–704.

Fan, D.-P.; Ji, G.-P.; Sun, G.; Cheng, M.-M.; Shen, J.; and Shao, L. 2020a. Camouflaged Object Detection. In *CVPR*.

Fan, D.-P.; Ji, G.-P.; Zhou, T.; Chen, G.; Fu, H.; Shen, J.; and Shao, L. 2020b. PraNet: Parallel reverse attention network for polyp segmentation. In *MICCAI*, 263–273. Springer.

Fan, D.-P.; Zhai, Y.; Borji, A.; Yang, J.; and Shao, L. 2020c. BBS-Net: RGB-D salient object detection with a bifurcated backbone strategy network. In *ECCV*.

Feng, M.; Lu, H.; and Ding, E. 2019. Attentive Feedback Network for Boundary-aware Salient Object Detection. In *CVPR*.

He, K.; Zhang, X.; Ren, S.; and Jian, S. 2016. Deep Residual Learning for Image Recognition. In *CVPR*, 770–778.

Hou, Q.; Cheng, M.-M.; Hu, X.; Borji, A.; Tu, Z.; and Torr, P. H. 2017. Deeply supervised salient object detection with short connections. In *CVPR*, 3203–3212.

Le, T.; Nguyen, T. V.; Nie, Z.; Tran, M.; and Sugimoto, A. 2019. Anabranch network for camouflaged object segmentation. *Computer Vision and Image Understanding* 184: 45–56.

Li, G.; and Yu, Y. 2015. Visual Saliency Based on Multiscale Deep Features. In *CVPR*, 5455–5463.

Li, Y.; Hou, X.; Koch, C.; Rehg, J. M.; and Yuille, A. L. 2014. The Secrets of Salient Object Segmentation. In *CVPR*, 280–287.

Liu, J.-J.; Hou, Q.; Cheng, M.-M.; Feng, J.; and Jiang, J. 2019. A Simple Pooling-Based Design for Real-Time Salient Object Detection. In *CVPR*.

Long, J.; Shelhamer, E.; and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. In *CVPR*, 3431–3440.

Mottaghi, R.; Chen, X.; Liu, X.; Cho, N.; Lee, S.; Fidler, S.; Urtasun, R.; and Yuille, A. L. 2014. The Role of Context for Object Detection and Semantic Segmentation in the Wild 891–898.

Owens, A.; Barnes, C.; Flint, A.; Singh, H.; and Freeman, W. 2014. Camouflaging an Object from Many Viewpoints. In *CVPR*, 2782–2789.

Pang, Y.; Zhao, X.; Zhang, L.; and Lu, H. 2020. Multi-Scale Interactive Network for Salient Object Detection. In *CVPR*.

Perazzi, F.; Krahenbuhl, P.; Pritch, Y.; and Hornung, A. 2012. Saliency filters: Contrast based filtering for salient region detection. In *CVPR*, 733–740.

Rahman, M. A.; and Wang, Y. 2016. Optimizing Intersection-Over-Union in Deep Neural Networks for Image Segmentation. In *International Symposium on Visual Computing*.

Wang, L.; Lu, H.; Wang, Y.; Feng, M.; Wang, D.; Yin, B.; and Ruan, X. 2017. Learning to detect salient objects with image-level supervision. In *CVPR*, 136–145.

Wang, L.; Wang, L.; Lu, H.; Zhang, P.; and Ruan, X. 2019a. Salient Object Detection with Recurrent Fully Convolutional Networks. *IEEE TPAMI* 41(7): 1734–1746.

Wang, W.; Shen, J.; Dong, X.; Borji, A.; and Yang, R. 2019b. Inferring Salient Objects from Human Fixations. *IEEE TPAMI* .

Wang, W.; Zhao, S.; Shen, J.; Hoi, S. C. H.; and Borji, A. 2019c. Salient Object Detection With Pyramid Attention and Salient Edges. In *CVPR*, 1448–1457.

Wei, J.; Wang, S.; and Huang, Q. 2020. F^{3}Net: Fusion, Feedback and Focus for Salient Object Detection. In *AAAI*.

Wu, R.; Feng, M.; Guan, W.; Wang, D.; Lu, H.; and Ding, E. 2019. A Mutual Learning Method for Salient Object Detection With Intertwined Multi-Supervision. In *CVPR*, 8150–8159.

Wu, Z.; Su, L.; and Huang, Q. 2019a. Cascaded Partial Decoder for Fast and Accurate Salient Object Detection. In *CVPR*.

Wu, Z.; Su, L.; and Huang, Q. 2019b. Stacked Cross Refinement Network for Edge-Aware Salient Object Detection. In *ICCV*.

Xie, S.; and Tu, Z. 2015. Holistically-nested edge detection. In *ICCV*, 1395–1403.

Yan, Q.; Li, X.; Shi, J.; and Jia, J. 2013. Hierarchical Saliency Detection. In *CVPR*, 1155–1162.

Yang, C.; Zhang, L.; Lu, H.; Ruan, X.; and Yang, M. 2013. Saliency Detection via Graph-Based Manifold Ranking 3166–3173.

Zhang, J.; Fan, D.-P.; Dai, Y.; Anwar, S.; Saleh, F. S.; Zhang, T.; and Barnes, N. 2020. UC-Net: uncertainty inspired rgb-d saliency detection via conditional variational autoencoders. In *CVPR*, 8582–8591.

Zhang, J.; Sclaroff, S.; Lin, Z.; Shen, X.; Price, B.; and Mech, R. 2015. Minimum Barrier Salient Object Detection at 80 FPS. In *ICCV*, 1404–1412.

Zhao, J.-X.; Liu, J.-J.; Fan, D.-P.; Cao, Y.; Yang, J.; and Cheng, M.-M. 2019a. EGNet:Edge Guidance Network for Salient Object Detection. In *ICCV*.

Zhao, X.; Pang, Y.; Zhang, L.; Lu, H.; and Zhang, L. 2020. Suppress and Balance: A Simple Gated Network for Salient Object Detection. In *ECCV*.

Zhao, Z.; Zheng, P.; Xu, S.; and Wu, X. 2019b. Object Detection With Deep Learning: A Review. *IEEE Transactions on Neural Networks* 30(11): 3212–3232.

Zheng, Y.; Zhang, X.; Wang, F.; Cao, T.; Sun, M.; and Wang, X. 2019. Detection of People With Camouflage Pattern Via Dense Deconvolution Network. *IEEE Signal Processing Letters* 26(1): 29–33.

Zhou, H.; Xie, X.; Lai, J.-H.; Chen, Z.; and Yang, L. 2020. Interactive Two-Stream Decoder for Accurate and Fast Saliency Detection. In *CVPR*.

Zhu, W.; Liang, S.; Wei, Y.; and Sun, J. 2014. Saliency Optimization from Robust Background Detection 2814–2821.