Exploring Figure-Ground Assignment Mechanism in Perceptual Organization (Appendix)

Wei Zhai[1] **Yang Cao**[1,3*] **Jing Zhang**[2] **Zheng-Jun Zha**[1]
[1]University of Science and Technology of China [2]The University of Sydney
[3]Institute of Artificial Intelligence, Hefei Comprehensive National Science Center
{wzhai056,forrest,zhazj}@ustc.edu.cn, jing.zhang1@sydney.edu.au

Due to the page limit of the paper, we provide a more detailed description and experimental results in this appendix. The main content includes the following four folds: 1) The detailed instructions of constructing Figure-Ground Cues (in Section A). 2) We describe the definition of evaluation metrics used in the paper (in Section B). 3) The implementation details of IEM (in Section C). 4) The detailed description of the Figure-Ground Segregation Test (in Section D). 5) The details of the model implementation in the ablation experiment (in Section E). 6) We present more comparison results with other methods on three applications (in Section F). 7) We discuss the limitation of our proposed method (in Section G).

Contents

*Corresponding author.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Algorithm 1 Figure-Ground Cues Generation: Python-like Pseudocode

```
# "label" is the original label

label_o = label[:, :, 0]
# Generate Convexity Cue
# set kernel size
kernel = skimage.morphology.disk(10)
# opening
label_op = skimage.morphology.opening(label_o, kernel)
label_op = label_o - label_op
# closing
label_cl = skimage.morphology.closing(label_o, kernel)
label_cl = label_cl - label_o

label_opcl = np.zeros((2, label.shape[0], label.shape[1]), dtype=np.float32)
# Convexity label combine
label_opcl[0] = label_cl
label_opcl[1] = label_op

# Generate Lower Region Cue
# set kernel size
kernel = skimage.morphology.disk(5)
# erosion
label_er = skimage.morphology.erosion(label_o, kernel)
label_er = label_o - label_er
# dilation
label_di = skimage.morphology.dilation(label_o, kernel)
label_di = label_di - label_o

label_erdi = np.zeros((2, label.shape[0], label.shape[1]), dtype=np.float32)
# Lower Region label combine
label_erdi[0] = label_er
label_erdi[1] = label_di
```

A. Figure-Ground Assignment Cues

The figure-Ground assignment is the foundation of the visual perception process and contributes almost all perception-based tasks [1, 2]. The human visual mechanism points out that when humans observe images, they will use some configural cues to distinguish between foreground and background, including Convexity [3–6] and Lower Region [7, 8]. Accordingly, we propose to leverage the convexity and lower region labels, as shown in Figure S1. We use morphological opening/closing and morphological erosion/expansion operations to generate convexity and lower region labels. The key generation code for the Figure-Ground Cues is shown in the Algorithm 1.

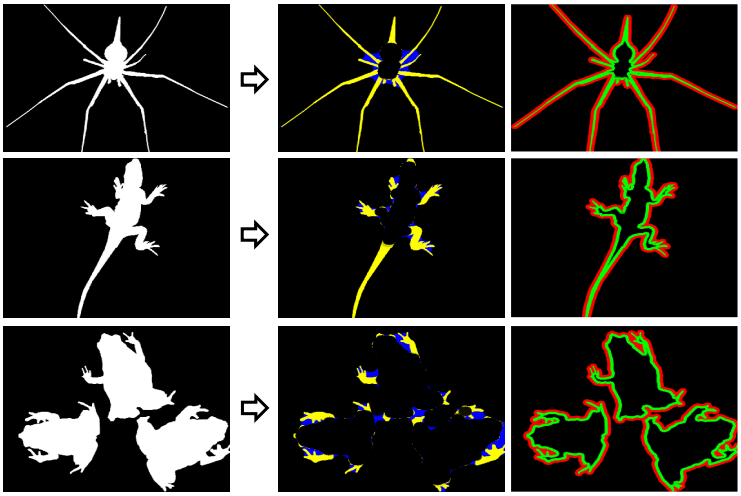


Figure S1: Some examples of Convexity and LR labels. “Convexity” represents the convexity label. “LR” represents the lower region label. Best viewed in color.

B. Evaluation Criteria

► **S-measure** (S_α) [9] focuses on evaluating the structural information of saliency maps, which is closer to the human visual system.

$$S_\alpha = \alpha S_o + (1-\alpha) S_r, \tag{1}$$

where S_o and S_r denotes the region-aware and object-aware structural similarity and α is set as 0.5 by default.

► **E-measure** (E_ϕ) [10] is designed for evaluating the difference between the predicted map X and ground-truth Y from the local and global perspectives.

$$E_\phi = \frac{1}{N} \sum_{n=1}^{N} \phi(X(n) - Y(n)), \tag{2}$$

where ϕ denotes the enhanced alignment matrix and n is the index of each pixel.

► **Weighted F-measure** (F_β^ω) [10] is an overall performance measurement and computed by the weighted harmonic mean of the precision and recall.

$$F_\beta^\omega = \frac{(1+\beta^2) Precision^\omega \times Recall^\omega}{\beta^2 \times Precision^\omega + Recall^\omega}, \tag{3}$$

where β^2 is set to 0.3 as in previous works.

► **Mean Absolute Error** (*MAE*) [11] measures the average absolute distance between the normalized predicted map and the ground-truth.

$$MAE = \frac{1}{N} \sum_{n=1}^{N} |X(n) - Y(n)|. \tag{4}$$

► **Dice** [12] coefficient is a similarity measure function, typically used to calculate the similarity of two samples, with values in the range $[0, 1]$.

$$Dice = \frac{2\,|X \cap Y|}{|X| + |Y|}. \tag{5}$$

where X is the predicted map and Y is the ground-truth.

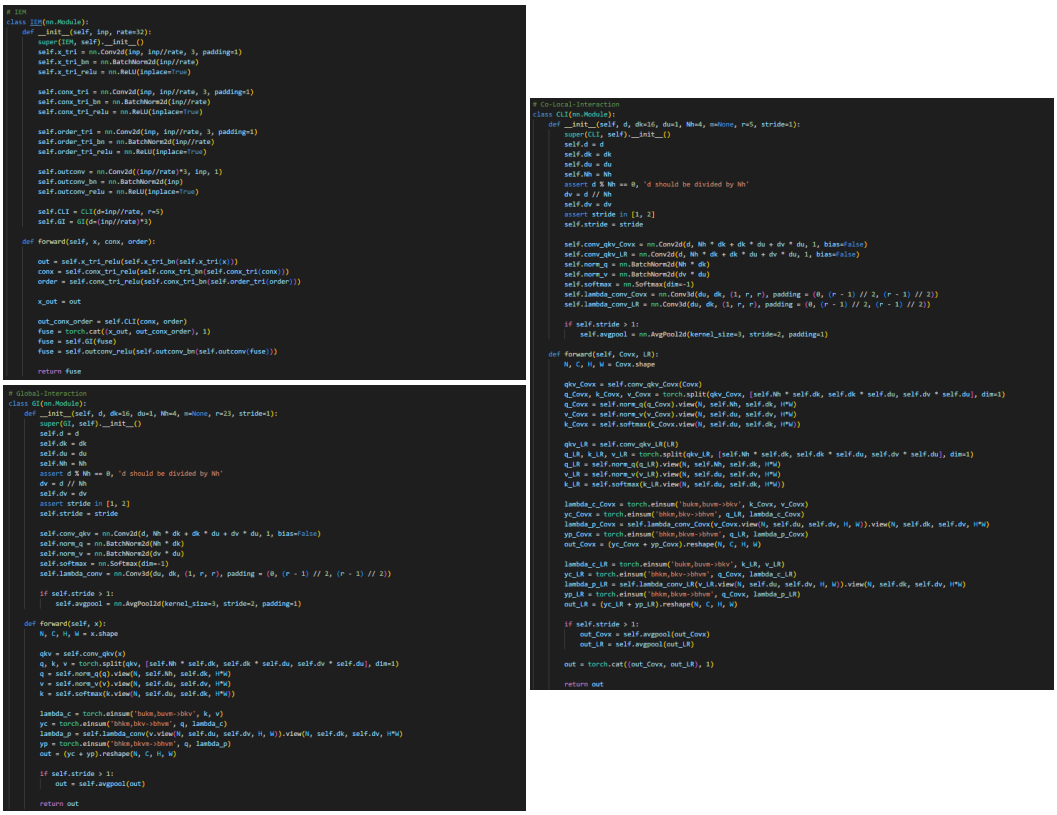
C. Implementation details of IEM

As shown in Figure S2, we provide the python source code about the IEM.

D. Figure-Ground Segregation Test

D.1. Dataset Generation

In this paper, to investigate the validity of our proposed method, we design and establish a set of tasks to evaluate the Figure-Ground assignment ability of deep convolutional neural networks, inspired by the Figure-Ground Segregation test in cognitive science experiments [13–16]. Furthermore, for a more comprehensive evaluation, three datasets of different difficulty levels (Easy, Normal, and Hard) are established by varying the transformations (as shown in Table S1), the division of texture image set, and the size of figure regions. The whole process of sample generation can be divided into the following steps: 1) Select a random image in the Pascal [17] dataset and use one object instance as a figure and the remaining regions as the ground. As shown in Table S1, we used the area of the object region as a controllable factor for the generation of dataset. 2) Given a collection of texture images (like DTD [18]), randomly select a texture image from it. The division of texture sets in different datasets is shown in Table S2. 3) Two random transformations (such as rotation and scaling) are performed on the texture image independently. 4) Fill the figure region and the ground region with the two transformed textures, respectively, resulting in a synthetic sample. The synthesis process is shown in Figure S3. The link to download the above datasets are: Pascal (http://host.robots.ox.ac.uk/pascal/VOC/) and DTD (https://www.robots.ox.ac.uk/ vgg/data/dtd/).



```python
# IEM
class IEM(nn.Module):
    def __init__(self, inp, rate=32):
        super(IEM, self).__init__()
        self.x_tri = nn.Conv2d(inp, inp//rate, 3, padding=1)
        self.x_tri_bn = nn.BatchNorm2d(inp//rate)
        self.x_tri_relu = nn.ReLU(inplace=True)

        self.conx_tri = nn.Conv2d(inp, inp//rate, 3, padding=1)
        self.conx_tri_bn = nn.BatchNorm2d(inp//rate)
        self.conx_tri_relu = nn.ReLU(inplace=True)

        self.order_tri = nn.Conv2d(inp, inp//rate, 3, padding=1)
        self.order_tri_bn = nn.BatchNorm2d(inp//rate)
        self.order_tri_relu = nn.ReLU(inplace=True)

        self.outconv = nn.Conv2d((inp//rate)*3, inp, 1)
        self.outconv_bn = nn.BatchNorm2d(inp)
        self.outconv_relu = nn.ReLU(inplace=True)

        self.CLI = CLI(d=inp//rate, r=5)
        self.GI = GI(d=(inp//rate)*3)

    def forward(self, x, conx, order):

        out = self.x_tri_relu(self.x_tri_bn(self.x_tri(x)))
        conx = self.conx_tri_relu(self.conx_tri_bn(self.conx_tri(conx)))
        order = self.conx_tri_relu(self.conx_tri_bn(self.order_tri(order)))

        x_out = out

        out_conx_order = self.CLI(conx, order)
        fuse = torch.cat((x_out, out_conx_order), 1)
        fuse = self.GI(fuse)
        fuse = self.outconv_relu(self.outconv_bn(self.outconv(fuse)))

        return fuse
```

```python
# Global-Interaction
class GI(nn.Module):
    def __init__(self, d, dk=16, du=1, Nh=4, m=None, r=23, stride=1):
        super(GI, self).__init__()
        self.d = d
        self.dk = dk
        self.du = du
        self.Nh = Nh
        assert d % Nh == 0, 'd should be divided by Nh'
        dv = d // Nh
        self.dv = dv
        assert stride in [1, 2]
        self.stride = stride

        self.conv_qkv = nn.Conv2d(d, Nh * dk + dk * du + dv * du, 1, bias=False)
        self.norm_q = nn.BatchNorm2d(Nh * dk)
        self.norm_v = nn.BatchNorm2d(dv * du)
        self.softmax = nn.Softmax(dim=-1)
        self.lambda_conv = nn.Conv3d(du, dk, (1, r, r), padding = (0, (r - 1) // 2, (r - 1) // 2))

        if self.stride > 1:
            self.avgpool = nn.AvgPool2d(kernel_size=3, stride=2, padding=1)

    def forward(self, x):
        N, C, H, W = x.shape

        qkv = self.conv_qkv(x)
        q, k, v = torch.split(qkv, [self.Nh * self.dk, self.dk * self.du, self.dv * self.du], dim=1)
        q = self.norm_q(q).view(N, self.Nh, self.dk, H*W)
        v = self.norm_v(v).view(N, self.du, self.dv, H*W)
        k = self.softmax(k.view(N, self.du, self.dk, H*W))

        lambda_c = torch.einsum('bukm,buvm->bkv', k, v)
        yc = torch.einsum('bhkm,bkv->bhvm', q, lambda_c)
        lambda_p = self.lambda_conv(v.view(N, self.du, self.dv, H, W)).view(N, self.dk, self.dv, H*W)
        yp = torch.einsum('bhkm,bkvm->bhvm', q, lambda_p)
        out = (yc + yp).reshape(N, C, H, W)

        if self.stride > 1:
            out = self.avgpool(out)

        return out
```

```python
# Co-local-interaction
class CLI(nn.Module):
    def __init__(self, d, dk=16, du=1, Nh=4, m=None, r=5, stride=1):
        super(CLI, self).__init__()
        self.d = d
        self.dk = dk
        self.du = du
        self.Nh = Nh
        assert d % Nh == 0, 'd should be divided by Nh'
        dv = d // Nh
        self.dv = dv
        assert stride in [1, 2]
        self.stride = stride

        self.conv_qkv_Covx = nn.Conv2d(d, Nh * dk + dk * du + dv * du, 1, bias=False)
        self.conv_qkv_LR = nn.Conv2d(d, Nh * dk + dk * du + dv * du, 1, bias=False)
        self.norm_q = nn.BatchNorm2d(Nh * dk)
        self.norm_v = nn.BatchNorm2d(dv * du)
        self.softmax = nn.Softmax(dim=-1)
        self.lambda_conv_Covx = nn.Conv3d(du, dk, (1, r, r), padding = (0, (r - 1) // 2, (r - 1) // 2))
        self.lambda_conv_LR = nn.Conv3d(du, dk, (1, r, r), padding = (0, (r - 1) // 2, (r - 1) // 2))

        if self.stride > 1:
            self.avgpool = nn.AvgPool2d(kernel_size=3, stride=2, padding=1)

    def forward(self, Covx, LR):
        N, C, H, W = Covx.shape

        qkv_Covx = self.conv_qkv_Covx(Covx)
        q_Covx, k_Covx, v_Covx = torch.split(qkv_Covx, [self.Nh * self.dk, self.dk * self.du, self.dv * self.du], dim=1)
        q_Covx = self.norm_q(q_Covx).view(N, self.Nh, self.dk, H*W)
        v_Covx = self.norm_v(v_Covx).view(N, self.du, self.dv, H*W)
        k_Covx = self.softmax(k_Covx.view(N, self.du, self.dk, H*W))

        qkv_LR = self.conv_qkv_LR(LR)
        q_LR, k_LR, v_LR = torch.split(qkv_LR, [self.Nh * self.dk, self.dk * self.du, self.dv * self.du], dim=1)
        q_LR = self.norm_q(q_LR).view(N, self.Nh, self.dk, H*W)
        v_LR = self.norm_v(v_LR).view(N, self.du, self.dv, H*W)
        k_LR = self.softmax(k_LR.view(N, self.du, self.dk, H*W))

        lambda_c_Covx = torch.einsum('bukm,buvm->bkv', k_Covx, v_Covx)
        yc_Covx = torch.einsum('bhkm,bkv->bhvm', q_LR, lambda_c_Covx)
        lambda_p_Covx = self.lambda_conv_Covx(v_Covx.view(N, self.du, self.dv, H, W)).view(N, self.dk, self.dv, H*W)
        yp_Covx = torch.einsum('bhkm,bkvm->bhvm', q_LR, lambda_p_Covx)
        out_Covx = (yc_Covx + yp_Covx).reshape(N, C, H, W)

        lambda_c_LR = torch.einsum('bukm,buvm->bkv', k_LR, v_LR)
        yc_LR = torch.einsum('bhkm,bkv->bhvm', q_Covx, lambda_c_LR)
        lambda_p_LR = self.lambda_conv_LR(v_LR.view(N, self.du, self.dv, H, W)).view(N, self.dk, self.dv, H*W)
        yp_LR = torch.einsum('bhkm,bkvm->bhvm', q_Covx, lambda_p_LR)
        out_LR = (yc_LR + yp_LR).reshape(N, C, H, W)

        if self.stride > 1:
            out_Covx = self.avgpool(out_Covx)
            out_LR = self.avgpool(out_LR)

        out = torch.cat((out_Covx, out_LR), 1)

        return out
```

Figure S2: The python source code about the IEM.

Table S1: Details of transformation parameters used in Figure-Ground Segregation Test."Texture split" means the number of texture classes used in the training and testing sets. In this paper, we use the DTD [18] dataset as a candidate texture set, which includes 47 classes of textures (the detail is shown in Table S2). "Rotation control" indicates the rotation angle applied to the foreground and background textures. "Area control" represents the size of foreground area when selecting foregrounds from the pascal dataset. Using $\geqslant 5000$ as an example, we only select the area of the object with a foreground area greater than 5000 pixels.

$(train : test)$	**Easy**	**Normal**	**Hard**
Texture split	$40 : 7$	$10 : 37$	$3 : 44$
Rotation control	[1°∼90°]:[1°∼90°]	[30°∼90°]:[1°∼60°]	[60°∼90°]:[1°∼30°]
Area control	$\geqslant 5000$	$\geqslant 2500$	$\geqslant 500$

D.2. Detail Description of Strong Baseline

In this paper, we compare our model with two representative architectures: U-Net [19] and Deeplabv3 [20]. U-Net is one of the most popular architectures in the field of segmentation, which has a typical encoder-decoder architecture, and the shallow features are directly fed into the decoder through skip connections. Deeplabv3 is a highly competitive network for semantic segmentation. Through atrous spatial pyramid pooling, it can expand the receptive field while modeling fine details. Furthermore, we also compare the results of the two models after adding edge supervision. Their detailed structures are shown in Figure S4.

D.3. Detailed Results

Due to the page limit of the paper, we give the numerical results in this section (as shown in Table S3 and Table S4). In addition, we show more visual results. We compare the performance of our method

Table S2: Different texture classes are used in the training and testing sets at different difficulty levels.

	Train	Test
Easy	bumpy, meshed, porous, perforated, stained, grid, braided, blotchy, lacelike, pitted, bubbly, wrinkled, fibrous, chequered, marbled, paisley, flecked, waffled, pleated, striped, cobwebbed, stratified, honeycombed, studded, polka-dotted, smeared, dotted, zigzagged, matted, spiralled, banded, cracked, swirly, lined, crystalline, freckled, frilly, gauzy, interlaced, scaly	sprinkled, potholed, knitted, veined, woven, grooved, crosshatched
Normal	bumpy, meshed, porous, perforated, stained, grid, braided, blotchy, lacelike, pitted	bubbly, wrinkled, fibrous, chequered, marbled, paisley, flecked, waffled, pleated, striped, cobwebbed, stratified, honeycombed, studded, polka-dotted, smeared, dotted, zigzagged, matted, spiralled, banded, cracked, swirly, lined, crystalline, freckled, frilly, gauzy, interlaced, scaly, sprinkled, potholed, knitted, veined, woven, grooved, crosshatched
Hard	bumpy, meshed, porous	perforated, stained, grid, braided, blotchy, lacelike, pitted, bubbly, wrinkled, fibrous, chequered, marbled, paisley, flecked, waffled, pleated, striped, cobwebbed, stratified, honeycombed, studded, polka-dotted, smeared, dotted, zigzagged, matted, spiralled, banded, cracked, swirly, lined, crystalline, freckled, frilly, gauzy, interlaced, scaly, sprinkled, potholed, knitted, veined, woven, grooved, crosshatched

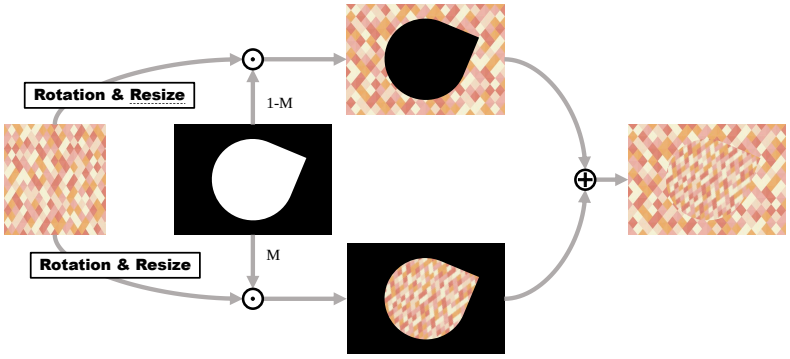
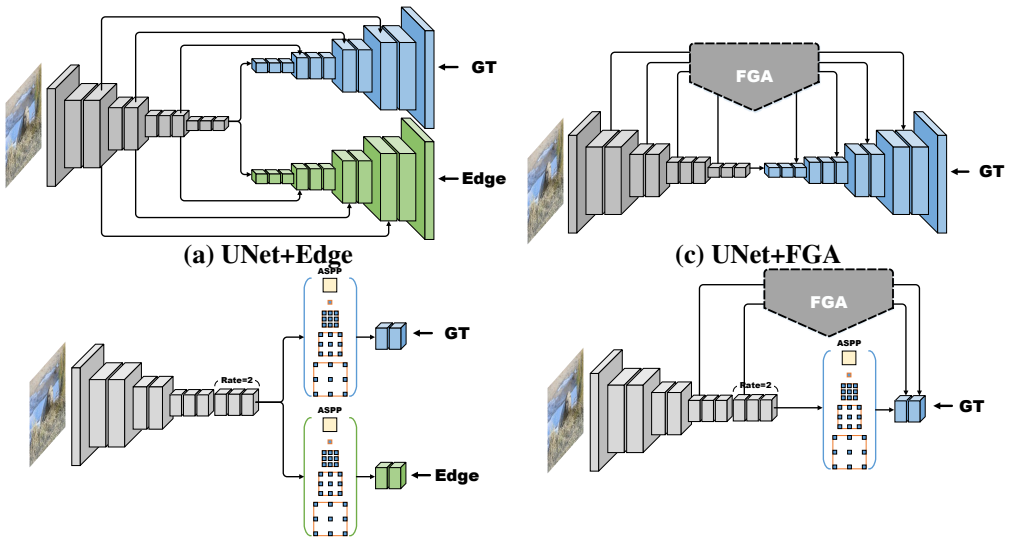


Figure S3: The generation process of the sample in Figure-Ground Segregation Test.

with several representative segmentation methods for the Figure-Ground Segregation Test, as shown in Figure S5.

D.4. More analysis

We explore the relevant hyperparameters of IEM on the normal dataset of figure-ground segregation test. We explore the relative positional embedding range of the CLI part of the IEM, a parameter that controls the scale of the IEM for modeling local contexts. As shown in the Table S5, the performance improves further as the relative position embedding range increases, but the magnitude of the improvement becomes diminished. Moreover, to further reduce the number of parameters and computation in



(a) UNet+Edge (c) UNet+FGA

(b) DeepLabv3+Edge (d) DeepLabv3+FGA

Figure S4: Detailed structures of UNet+Edge/UNet+FGA and Deeplabv3+Edge/Deeplabv3+FGA. The different colors represent different weights.

Table S3: FGA-Net outperforms representative models for Figure-Ground Segregation. This table corresponds to Figure 6 (a) in the body text.

	UNet		UNet+Edge		UNet+FGA		DeepLabv3		DeepLabv3+Edge		DeepLab+FGA	
Metrics	Sm	IoU	Sm	IoU	Sm	IoU	Sm	IoU	Sm	IoU	Sm	IoU
Easy	0.866	0.793	0.878	0.801	0.883	0.811	0.875	0.796	0.878	0.805	0.899	0.82
Normal	0.741	0.573	0.755	0.602	0.816	0.659	0.774	0.608	0.782	0.64	0.825	0.671
Hard	0.648	0.371	0.662	0.396	0.733	0.472	0.664	0.412	0.679	0.433	0.758	0.501

IEM, we incorporate a bottleneck strategy similar to that in ResBlock, where we explore the impact of the scaling ratio of bottlenecks in IEM on performance (Table S6). Experimental results show that the smaller the scaling ratio (indicating a larger number of parameters), the performance can be improved, but the improvement is limited.

We perform an experiment on the normal dataset of the figure-ground segregation test, and the experimental result (Table S7) shows that directly using an edge-aware loss like HED [21] instead of $L_L R$ does not achieve better result than the FGA model itself. The reason is that the supervision using edge-aware only provides the perception of the foreground boundary, while the Lower Region cue provides a more contextual perception of the boundary region beyond the foreground boundary. In brief, it allows the model to focus on the associations within the inner and outer regions of the boundary and the differences between the inner and outer regions.

E. Details of the model implementation in the ablation experiment

The detailed structure of model (a)(b) and (c) in the ablation experiment is shown in Figure S6.

F. More Results on Three Challenging Applications

F.1. More Sample Images from Three Challenging Applications

In this paper, we verify the effectiveness and superiority of our proposed framework on three challenging segmentation tasks: Camouflaged Object Detection (COD) [22], Polyp Segmentation (PS) [23], and Lung Infection Segmentation (LIS) [24]. These three tasks all show high similarity between the foreground object and the background region. In addition, they have some unique characteristics. COD task mainly manifests in complex background/edges, small targets/structure,

Table S4: FGA-Net is more data-efficient than representative models. The table corresponds to Figure 6 (b) in the body text.

IoU	1 %	10 %	30 %	100 %
Baseline	0.202	0.375	0.543	0.602
+Edge	0.245	0.415	0.572	0.64
+FGA	0.375	0.508	0.601	0.659

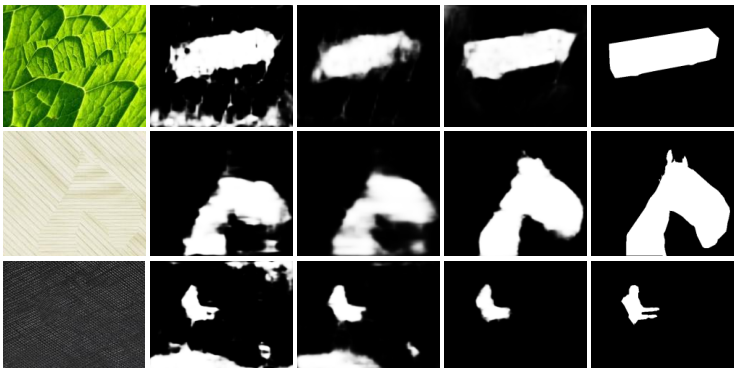


Figure S5: Visual comparisons on the Normal dataset.

slender trunks/limbs, and partially occluded objects. The main challenge of the PS task is the diversity of size and texture. The main difficulty in the LIS task is the complex distribution of foregrounds and backgrounds. To show the characteristics of different tasks, we give some sample images from these three challenging tasks (Figure S7). The link to download the above datasets are: COD (http://dpfan.net/Camouflage/), PS (https://github.com/DengPingFan/PraNet), and LIS (https://medicalsegmentation.com/covid19/).

F.2. Results on Camouflaged Object Detection Task

Visual Comparisons. We show qualitative results in this section. We compare the performance of our method with several state-of-the-art (SOTA) methods on CHAMELEON [25], CAMO [26], and COD10K [22] datasets. All the methods are listed as follows: UNet++ [27], PiCANet [28], BASNet [29], PFANet [30], CPD [31], EGNet [32], and SINet [22]. As shown in Figure S8, our FGA-Net is robust in dealing with various challenging scenarios, including small object, multiple objects, low contrast and complex background.

Ablation. We add the relevant ablation experiments on the realistic dataset (COD), as shown in Table S8 and Table S9, the performance is generally consistent with that in the Figure-Ground Segregation Test.

Analysis. To verify the effectiveness of our method, we visualize the output of FGA-Net in Figure S9. It can be seen that in addition to the prediction map output by the main branch, the prediction map output by the other two branches is also close to GT. We further visualize the feature maps to verify the effectiveness of our proposed FGA framework. By comparing the left part (w/o FGA) with the right part (w/ FGA), feature maps after the aggregation module are more complete in object regions than those without the FGA framework.

F.3. Results on Polyp Segmentation Task

We show qualitative results in this section. We compare the performance of our method with several SOTA methods on Kvasir **(Kvasir)** [34], CVC-ClinicDB/CVC-612 **(CVC-612)** [35], CVC-ColonDB **(ColonDB)** [36], ETIS **(ETIS)** [37], and EndoScene **(Endo)** [38] datasets, as shown in Figure S10.

Table S5: The relative positional embedding range (r) w.r.t performance.

r	3	5	7	9
S	0.813	0.816	0.818	0.818
IoU	0.690	0.695	0.696	0.698

Table S6: The scaling ratio (rate) of bottlenecks in IEM w.r.t performance.

rate	32	16	8
S	0.816	0.819	0.819
IoU	0.695	0.700	0.703

F.4. Results on Lung Infection Segmentation Task

We show qualitative results in this section. We compare the performance of our method with several SOTA methods on COVID-19 datasets. As shown in Figure S11.

G. Limitation

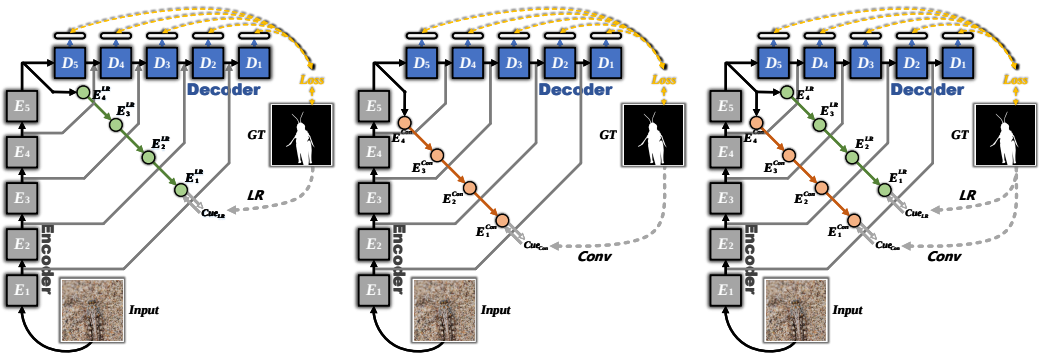
Figure S12 shows a sample failure about our approach. Our method produces inaccurate result in the case of excessively complex foreground content. This is probably due to the fact that our FGA model learns configural statistics that do not contain these local trivial details. To cope with this situation, we can add specific network [29] or post-processing measures [41] used to refine the details behind the existing model.

Our proposed method shows promising results when only a single or few object instances are in the scene. Still, the problem of segmentation errors occurs when there are many multiple separated objects in the scene, which limits its application to some extent. To extend the application of FGA, we later want to introduce it into the segmentation part of the Mask-RCNN-based method [42, 43].

Societal Impact. Perceptual organization is one of the most challenging computer vision tasks. Albeit being challenging, Perceptual organization is beneficial to a wide range of applications. However, there also exists the risk that the technology is utilized in the scenario of the illegal shoot, malicious edit, and incorrect use.

Table S7: Directly using an edge-aware loss like HED [21] instead of $L_L R$.

	FGA	FGA (HED)
S	0.816	0.807
IoU	0.695	0.642



model (a) **model (b)** **model (c)**

Figure S6: The detailed structure of model (a)(b) and (c) in the ablation experiment.

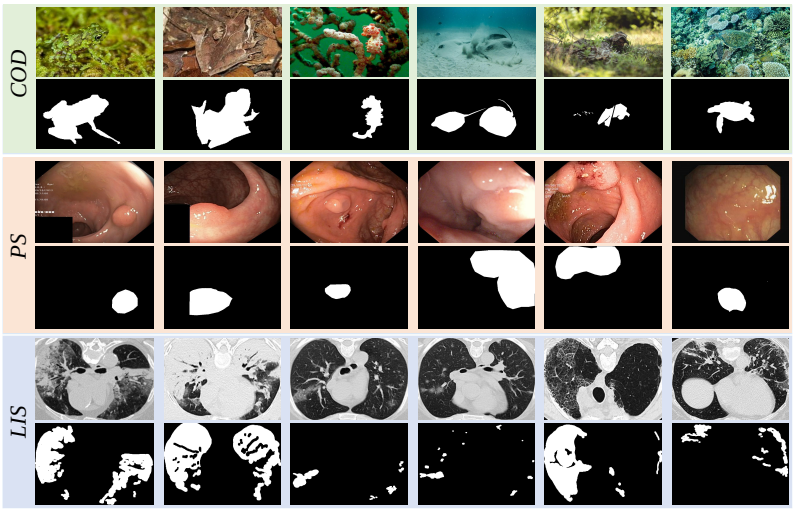


Figure S7: Sample images from three challenging applications.

Table S8: Ablation study of the proposed FGA-Net on COD dataset.

	LR	Covx	IEM	S	E	F	M
Baseline				.795	.878	.649	.063
(a)	✓			.808	.871	.669	.040
(b)		✓		.803	.866	.654	.039
(c)	✓	✓		.813	.880	.683	.035
FGA	✓	✓	✓	.821	.895	.687	.031

Table S9: Ablation study of IEM on COD dataset.

Local	Collaborative	Global	Lambda	S_α	IoU	S_α	IoU
✓				.815	.884	.680	.037
✓	✓			.816	.889	.690	.036
		✓		.813	.880	.681	.037
✓	✓	✓		.818	.890	.686	.033
✓	✓	✓	✓	.821	.895	.687	.031

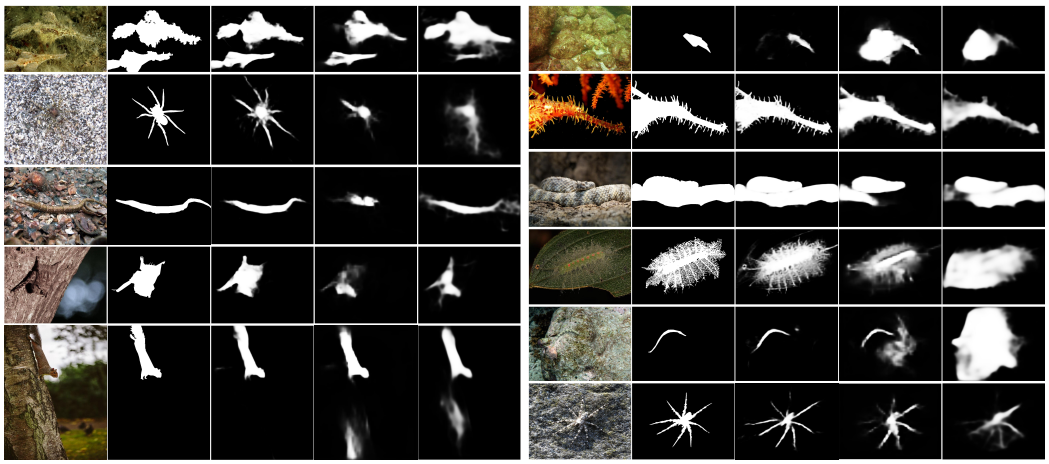


Figure S8: Visual comparisons of our FGA-Net and SOTA methods on COD task. Compared with other methods, our method can detect the overall camouflaged object and refine the details of the detected camouflaged object.

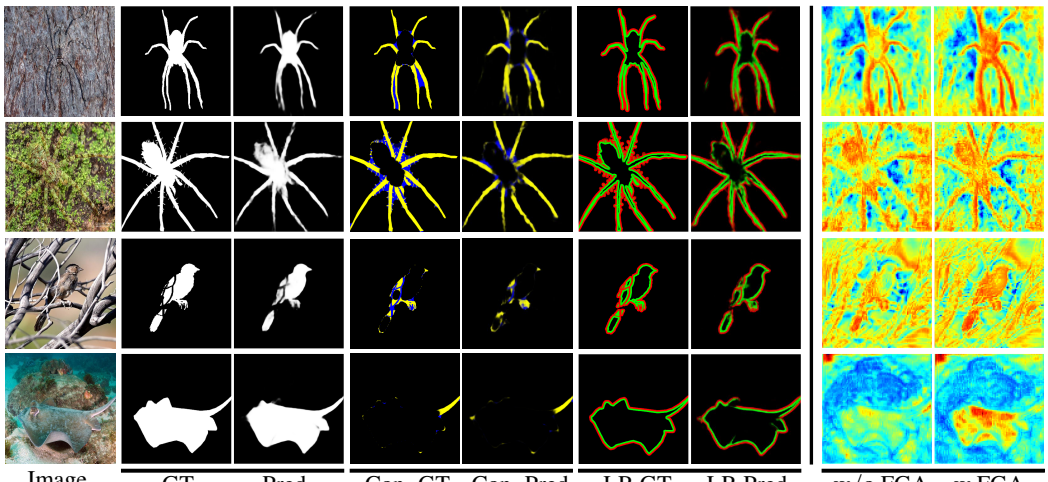


Figure S9: Visualization of our FGA-Net. "Con." represents convexity cue. "LR" represents lower region cue. "Pred." represents prediction map. "w/o FG" means the result of the baseline model.

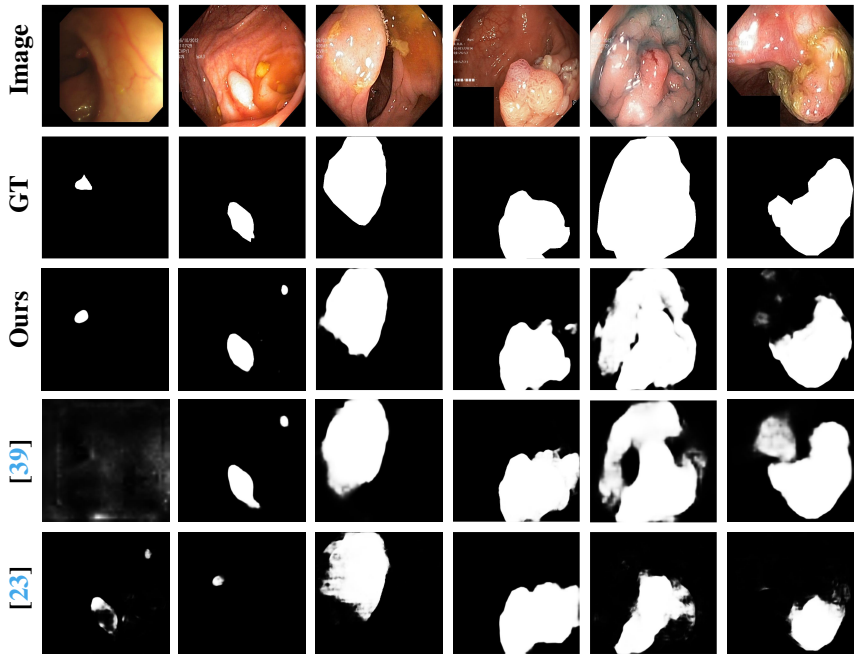


Figure S10: Visual comparisons of our FGA-Net and SOTA methods on PS task.

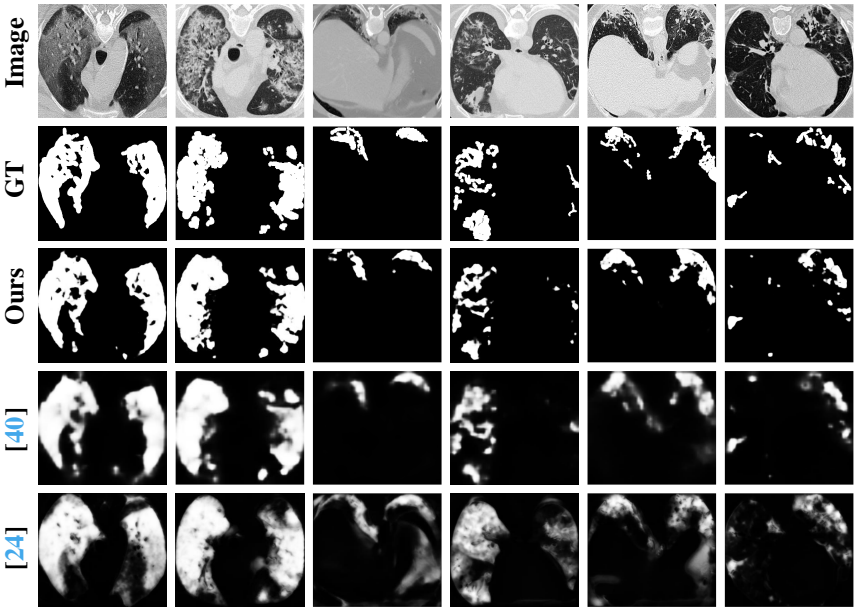


Figure S11: Visual comparisons of our FGA-Net and SOTA methods on LIS task.

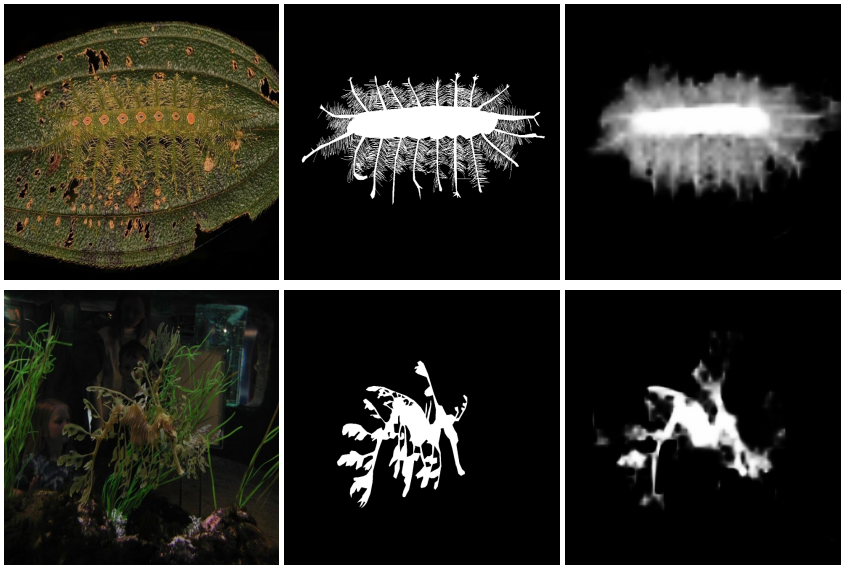


Image **GT** **Ours**

Figure S12: Failure case.

References

[1] Wagemans, J., J. H. Elder, M. Kubovy, et al. A century of gestalt psychology in visual perception: I. perceptual grouping and figure–ground organization. *Psychological bulletin*, 138(6):1172, 2012.

[2] Wagemans, J., J. Feldman, S. Gepshtein, et al. A century of gestalt psychology in visual perception: Ii. conceptual and theoretical foundations. *Psychological bulletin*, 138(6):1218, 2012.

[3] Metzger, W. Gesetze des sehens [laws of vision]. *Frankfurt: Waldemar Kramer*, 1953.

[4] Kanizsa, G. Convexity and symmetry in figure-ground organization. *Vision and artifact*, 1976.

[5] Peterson, M. A., E. Salvagio. Inhibitory competition in figure-ground perception: Context and convexity. *Journal of Vision*, 2008.

[6] Lu, Y., W. Zhang, H. Lu, et al. Salient object detection using concavity context. In *2011 International Conference on Computer Vision*. IEEE, 2011.

[7] Vecera, S. P., E. K. Vogel, G. F. Woodman. Lower region: A new cue for figure-ground assignment. *Journal of Experimental Psychology: General*, 2002.

[8] Fowlkes, C. C., D. R. Martin, J. Malik. Local figure-ground cues are valid for natural images. *Journal of Vision*, 7(8), 2007.

[9] Fan, D.-P., M.-M. Cheng, Y. Liu, et al. Structure-measure: A new way to evaluate foreground maps. In *IEEE International Conference on Computer Vision (ICCV)*. 2017.

[10] Margolin, R., L. Zelnik-Manor, A. Tal. How to evaluate foreground maps? In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014.

[11] Perazzi, F., P. Krähenbühl, Y. Pritch, et al. Saliency filters: Contrast based filtering for salient region detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[12] Milletari, F., N. Navab, S.-A. Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. IEEE, 2016.

[13] Lamme, V. A. The neurophysiology of figure-ground segregation in primary visual cortex. *Journal of neuroscience*, 15(2):1605–1615, 1995.

[14] Poort, J., F. Raudies, A. Wannig, et al. The role of attention in figure-ground segregation in areas v1 and v4 of the visual cortex. *Neuron*, 75(1):143–156, 2012.

[15] Roelfsema, P. R., V. A. Lamme, H. Spekreijse, et al. Figure—ground segregation in a recurrent network architecture. *Journal of cognitive neuroscience*, 14(4):525–537, 2002.

[16] Self, M. W., T. van Kerkoerle, H. Supèr, et al. Distinct roles of the cortical layers of area v1 in figure-ground segregation. *Current Biology*, 23(21):2121–2129, 2013.

[17] Everingham, M., S. A. Eslami, L. Van Gool, et al. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

[18] Cimpoi, M., S. Maji, I. Kokkinos, et al. Describing textures in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3606–3613. 2014.

[19] Ronneberger, O., P. Fischer, T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015.

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

[21] Xie, S., Z. Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403. 2015.

[22] Fan, D.-P., G.-P. Ji, G. Sun, et al. Camouflaged object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[23] Fan, D.-P., G.-P. Ji, T. Zhou, et al. Pranet: Parallel reverse attention network for polyp segmentation. 2020.

[24] Fan, D.-P., T. Zhou, G.-P. Ji, et al. Inf-net: Automatic covid-19 lung infection segmentation from ct images. *IEEE transactions on medical imaging*, 2020.

[25] Skurowski, P., H. Abdulameer, J. Błaszczyk, et al. Animal camouflage analysis: Chameleon database. *Unpublished Manuscript*, 2018.

[26] Le, T.-N., T. V. Nguyen, Z. Nie, et al. Anabranch network for camouflaged object segmentation. *Computer Vision and Image Understanding (CVIU)*, 2019.

[27] Zhou, Z., M. M. R. Siddiquee, N. Tajbakhsh, et al. Unet++: A nested u-net architecture for medical image segmentation. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. Springer, 2018.

[28] Liu, N., J. Han, M.-H. Yang. Picanet: Learning pixel-wise contextual attention for saliency detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[29] Qin, X., Z. Zhang, C. Huang, et al. Basnet: Boundary-aware salient object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[30] Zhao, T., X. Wu. Pyramid feature attention network for saliency detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[31] Wu, Z., L. Su, Q. Huang. Cascaded partial decoder for fast and accurate salient object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[32] Zhao, J.-X., J.-J. Liu, D.-P. Fan, et al. Egnet: Edge guidance network for salient object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[33] Fan, D.-P., G.-P. Ji, M.-M. Cheng, et al. Concealed object detection. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[34] Jha, D., P. H. Smedsrud, M. A. Riegler, et al. Kvasir-seg: A segmented polyp dataset. In *International Conference on Multimedia Modeling*. Springer, 2020.

[35] Bernal, J., F. J. Sánchez, G. Fernández-Esparrach, et al. Wm-dova maps for accurate polyp highlighting in colonoscopy: Validation vs. saliency maps from physicians. *Computerized Medical Imaging and Graphics*, 2015.

[36] Tajbakhsh, N., S. R. Gurudu, J. Liang. Automated polyp detection in colonoscopy videos using shape and context information. *IEEE transactions on medical imaging*, 2015.

[37] Silva, J., A. Histace, O. Romain, et al. Toward embedded detection of polyps in wce images for early diagnosis of colorectal cancer. *International Journal of Computer Assisted Radiology and Surgery*, 2014.

[38] Vázquez, D., J. Bernal, F. J. Sánchez, et al. A benchmark for endoluminal scene segmentation of colonoscopy images. *Journal of healthcare engineering*, 2017.

[39] Zhao, X., L. Zhang, H. Lu. Automatic polyp segmentation via multi-scale subtraction network. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 120–130. Springer, 2021.

[40] Ji, G.-P., L. Zhu, M. Zhuge, et al. Fast camouflaged object detection via edge-based reversible re-calibration network. *Pattern Recognition*, page 108414, 2021.

[41] Krähenbühl, P., V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *Advances in neural information processing systems*, 24, 2011.

[42] He, K., G. Gkioxari, P. Dollár, et al. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969. 2017.

[43] Huang, Z., L. Huang, Y. Gong, et al. Mask scoring r-cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6409–6418. 2019.