SRN: Side-Output Residual Network for Object Reflection Symmetry Detection and Beyond

Wei Ke, *Member, IEEE*, Jie Chen, *Member, IEEE*, Jianbin Jiao, *Member, IEEE*, Guoying Zhao, *Senior Member, IEEE*, and Qixiang Ye, *Senior Member, IEEE*

***Abstract*—This article establishes a baseline for object reflection symmetry detection in natural images by releasing a new benchmark named Sym-PASCAL and proposing an end-to-end deep learning approach for reflection symmetry. Sym-PASCAL spans challenges of multiobjects, object diversity, part invisibility, and clustered backgrounds, which is far beyond those in existing data sets. The end-to-end deep learning approach, referred to as a side-output residual network (SRN), leverages the output residual units (RUs) to fit the errors between the symmetry ground truth and the side outputs of multiple stages of a trunk network. By cascading RUs from deep to shallow, SRN exploits the "flow" of errors along multiple stages to effectively matching object symmetry at different scales and suppress the clustered backgrounds. SRN is interpreted as a boosting-like algorithm, which assembles features using RUs during network forward and backward propagations. SRN is further upgraded to a multitask SRN (MT-SRN) for joint symmetry and edge detection, demonstrating its generality to image-to-mask learning tasks. Experimental results verify that the Sym-PASCAL benchmark is challenging related to real-world images, SRN achieves state-of-the-art performance, and MT-SRN has the capability to simultaneously predict edge and symmetry mask without loss of performance.**

***Index Terms*—Edge detection, multitask deep learning, object reflection symmetry detection, side-output residual network (SRN).**

I. Introduction

REFLECTION symmetry is an inherent visual property of natural objects. With reflection symmetry, objects can be abstracted into descriptive and interpretable curves. Such curves constitute a continuous decomposition of object shapes [1], [2], providing discriminative cues for object representation. Reflection symmetry has been used to produce features and/or spatial constraints and applied to image segmentation [3], foreground extraction [4], object proposal [5], object detection [6], and text-line detection [7].

Early symmetry detection, often referred to as skeleton extraction, involves binary images [8], [9]. In recent years, symmetry detection has addressed processing color images [10]–[12], which uses cropped image patches focusing on the foreground. SYMMAX [13], WH-SYMMAX [14], SK506 [15], and SK-LARGE [16] are data sets with natural images. However, they lack either object-level annotation or in-the-wild settings.

To make object reflection symmetry close to practical application, we present a new data set with clustered backgrounds named Sym-PASCAL and an end-to-end deep symmetry detection approach. The new benchmark is derived from the PASCAL-VOC-2011 [17] segmentation data set, which is composed of 1435 natural images with 1742 individual objects. The data set is more challenging than the existing ones as multiple objects in a single image; object with different illuminations, viewpoints, and partially occluded; and the contextually cluttered scenes.

The object reflection symmetry detection is treated as an image-to-mask learning task. We propose a deep side-output residual network (SRN) for symmetry mask prediction. It uses color images as input and directly outputs the mask of object reflection symmetry. SRN is built on the successful holistically nested edge detection network (HED) [18] and upgrades it by cascading multiple residual units (RUs) on the side outputs, which are computed on different stages of convolutional layers with the fully convolutional network [19]. The RU is used to fit the error between the symmetry ground truth and the outputs of different convolutional stages. It is computationally easier as it pursues the minimization of residuals among scales rather than combines multiscale features to fit the ground truth. SRN provides an effective way to model the association and complementary among multilayer features. The RUs we defined not only significantly improve the performance of the baseline HED but also solve the learning convergence problem left by HED.

SRN is composed of a trunk network for feature learning and a branch network for hierarchical side-output processing. When multiple branch networks are added to the trunk network, SRN is upgraded to a multitask SRN (MT-SRN), where the trunk network is shared by different image-to-mask tasks, and each branch network specifies a task. The usage

Manuscript received April 16, 2019; revised November 23, 2019 and April 1, 2020; accepted April 24, 2020. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 61836012, Grant 61771447, Grant 61671427, and Grant 61972217, in part by the Natural Science Foundation of Guangdong Province in China under Grant 2019B1515120049, in part by the Academy of Finland for ICT 2023 Project under Grant 328115, in part by Infotech Oulu, and in part by Business Finland. *(Corresponding author: Qixiang Ye.)*

Wei Ke is with Xi'an Jiaotong University, Xi'an 710049, China (e-mail: wei.ke@xjtu.edu.cn).

Jie Chen is with the School of Electronic and Computer Engineering, Peking University, Shenzhen 518055, China, also with the Peng Cheng Laboratory, Shenzhen 518055, China, and also with the Center for Machine Vision and Signal Analysis, University of Oulu, 90014 Oulu, Finland (e-mail: chenj@pcl.ac.cn).

Guoying Zhao is with the Center for Machine Vision and Signal Analysis, University of Oulu, 90014 Oulu, Finland (e-mail: guoying.zhao@oulu.fi).

Jianbin Jiao and Qixiang Ye are with the School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing 101408, China (e-mail: jiaojb@ucas.ac.cn; qxye@ucas.ac.cn).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2020.2994325

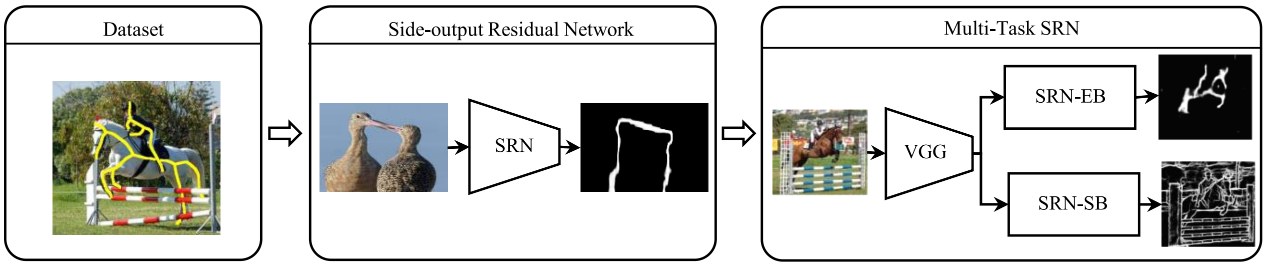


Fig. 1. Flowchart of the processes in this article. A data set is released in Section III, SRN is proposed in Section IV, and MT-SRN is introduced in Section V.

of a shared trunk network not only reduces the number of network parameters but also improves network learning by collectively leveraging training samples from multiple tasks. SRN was first proposed in our CVPR article [20] and is upgraded to MT-SRN in this full version. With extended experiments about symmetry detection and edge detection, the general applicability of SRN to image-to-mask learning tasks is further validated. The main contributions of this article are shown in Fig. 1, which includes the following:

1) an object reflection symmetry data set that spans challenges of diversity, multiobjects, part invisibility, and various clustered backgrounds, promoting the research of object symmetry to in-the-wild setting;
2) an SRN that fits the errors between ground truth and the outputs of different convolutional stages, enforcing representation capability under the principle of boosting-like feature ensemble;
3) an MT-SRN that performs joint symmetry and edge detection, showing not only the generality of SRN to image-to-mask learning tasks but also the feasibility to train a shared deep network on data sets for similar tasks.

A. Definition of Object Reflection Symmetry

An object is a reflection symmetric if it can be divided into two or more identical pieces that are arranged in an organized manner [10]. This means that an object is symmetric if there is a transformation that moves individual pieces of the object but does not change the overall shape. According to the way the pieces are organized and the type of transformation, an object has reflection, rotation, and scale symmetry, which, respectively, corresponds to the representation of skeleton, orientation, and scale. In this article, we involve reflection symmetry.

The reflection symmetry refers to the longest middle axis going through an object and dividing it into two pieces which are mirror images of each other. Formally, for an object with reflection shape, we divide it to homogeneous rectangles first, denoting as $\{R_i\}_{i=1}^N$, where N denotes the number of rectangles in the object, as shown in Fig. 2. For R_i, a segment of reflection symmetry, S_i, is defined as axis subject to

$$\begin{aligned} R_i(x, y) &\approx R_i(-x, y) \\ D_i(x, y) &= D_i(-x, y) \end{aligned} \tag{1}$$

where (x, y) denotes the coordinates in R_i, $(-x, y)$ denotes the symmetric coordinates in terms of the axis S_i, and $R_i(x, y)$ denotes the pixel value of point (x, y), as shown in Fig. 2. $D_i(x, y)$ and $D_i(-x, y)$, respectively, denote the distance

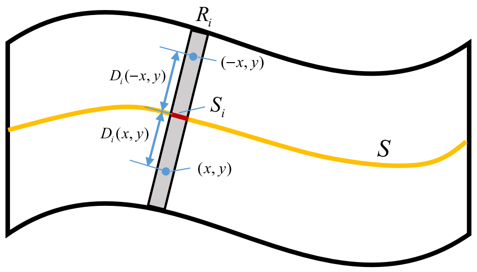


Fig. 2. Reflection symmetry refers to the longest middle axis (S) going through an object and dividing it into two pieces which are mirror images of each other.

between (x, y) or $(-x, y)$ and the symmetry S_i. Following the definition in [11], the first equation indicates that the pixel values are similar under S_i in the 2-D Euclidean space. The second equation implies that the distances of S_i to the two sides of object contours are the same.

When S_i tends to be infinitesimally small, the assembled set of S_i defines the object reflection symmetry, as

$$S = \{S_i\}_{i=1}^N. \tag{2}$$

The reminder of this article is organized as follows. Related work is reviewed in Section II. Section III introduces the Sym-PASCAL data set for in-the-wild object reflection symmetry detection. Sections IV and V detail the implementation of SRN and MT-SRN. Section VI presents experimental evaluation, and Section VII concludes this article.

II. Related Work

Symmetry has attracted much attention of the computer vision community. The benchmarks span from binary images [8], [21] to color images [10], [13]–[15], while the detection approaches range from handcrafted-based [22]–[24] to learning-based [3], [13]–[15], [20].

A. Benchmarks

Symmetry is qualitatively evaluated on quite limited binary shapes [8] for subjective perception or a few real-world images for objective evaluation [10], [25]. SYMMAX [13] contains hundreds of training and test images with reflection symmetry annotations, which can be regarded as an authentic benchmark. However, the reflection symmetry in SYMMAX is used for low-level image representation, regardless of objects.

WH-SYMMAX [14], SK506 [15], and SK-LARGE [16] are recently released benchmarks that are annotated with object skeleton. WH-SYMMAX is only composed of side-view horses, while SK506 and SK-LARGE consist of cropped objects from natural images. As none of them involves clustered background or multiple objects, plenty of room is left for us to develop new object reflection symmetry benchmarks. In the proposed Sym-PASCAL benchmark, we emphasize the challenging aspects, including multiobjects, object diversity, part invisibility, and various clustered backgrounds. We also provide Toolbox and instructions about how to create the data set and define criteria about how to determine whether an image contains multiple objects and/or partial invisibility.

B. Methods

1) Conventional Methods: In the early stage, symmetry extraction is known as skeleton detection[1] [9], [26], which is mainly proposed for binary images. Morphological operation is always used for these images. For color images, an instance segmentation step should be used as preprocessing, which is utilized to transfer color images to binary masks [21], [27]. As there is still no perfect solution for instance segmentation, integrating symmetry detection with instance segmentation not only accumulates the errors but also increases the complexity.

Extracting symmetry for color image based on superpixels is an alternative way [22]–[24]. Symmetry axes are extracted as a subset of lines that connect the center points for adjacent superpixels [22]. The symmetry curve can also be generated by the path of center point with a sequence of deformable disks within the superpixels [23]. In [24], the symmetry curve is linked with local skeleton segments with a particle filter, which models the smoothness and consistency of the symmetry.

Recently, learning-based methods are purposed for more effective symmetry detection in color images. A model is trained to predict whether a pixel is on the symmetry curve. Symmetry detection has seemed as an image-to-mask task with a dense prediction for all image pixels. Multiple instance learning is utilized for SYMMAX data set in [13], as the pixel is represented as an instance bag with multiorientation and multiscale. Structured random forest (SRF) [3] and subspace MIL [14] are employed to train a reflection symmetry detector with the same feature with [13]. Nevertheless, due to the limitation of the hand-designed features and conventional learning models, these methods are unable to detect the symmetry pixels with large scales, as much more context information needs to handle.

2) Deep Learning Methods: With the rise of deep learning, researchers have recently investigated the skeleton detection problem by fusing multilayer convolutional features [15], [20]. Symmetry detection is converted into an image-to-mask learning problem and unified with edge detection [18], [28]–[31] and semantic segmentation [19], [32], [33]. By using learned weights to combine the multilayer convolutional features, object skeletons/edges/segments are extracted in an end-to-end manner.

DeepContour [28] and HED [18] are two pioneering works that use learned weights to fuse the multiscale convolutional features and predict edges in an end-to-end manner. The trunk network of HED is an FCN [19], [34], which provides multiscale convolutional features for multiscale edge detection. When using the convolutional features in an oriented/encoder–decoder manner, HED is upgraded to detect oriented/higher level object contours [29], [30]. Bertasius *et al.* [31] further shown that they can predict object edges by exploiting object-level features from a multiscale side-output network.

FCN is also be applied to semantic segmentation in an image-to-mask manner, i.e., taking the input of arbitrary size images and producing the correspondingly sized output [19]. The application of FCN to semantic segmentation not only validates its applicability to spatially dense prediction tasks [34] but also contributes to multitask deep networks, e.g., the multitask network cascade [35] and joint object detection and semantic segmentation. In these works, multiscale convolutional features are usually used in a cascading manner, while the problems about how to minimize the errors among multiple scales and adaptively fit complex outputs with limited convolutional layers need to be further explored.

Most recently, a deep learning method, i.e., FSDS [15], is used to learn a model for skeleton predicting with the multiscale association on WH-SYMMAX and SK506 data sets. It is developed from HED [18] and supervises the side outputs with scale-associated ground truth. Even though it achieves good performance for symmetry detection, intensive annotation of scales for each symmetry point is needed, which takes more effort to prepare the data for training. Compared with FSDS, our proposed SRN matches the symmetry scale adaptively without requiring scale-associated annotation.

3) Multitask Learning: The concept of multitask learning is early proposed by Caruana [36]. Recently, multitask deep learning that aims to solve multiple tasks using a single network has attracted increased attention [37]–[40]. Sermanet *et al.* [37] used a CNN network for joint localization, detection, and classification. Eigen and Fergus [38] proposed a multiscale CNN for simultaneously predicting depth, surface normal, and semantic labels for an image. Eigen *et al.* [39] trained a network for joint person detection, pose estimation, and gender recognition. These methods have the advantage of performing multiple tasks with single deep networks but are difficult to be trained by diverse data sets.

In this article, SRN is extended to MT-SRN, which is trained on two diverse data sets. Compared with UberNet that fuses low-, mid-, and high-level vision tasks with diverse data sets, MT-SRN uses a different training strategy. It leverages a trunk network to extract low-level features, buffer convolutional layers to integrate the low-level features for the task-specific purpose, and SRNs to predict output on the buffer convolutional layers.

III. Object Symmetry Benchmark

A. Annotation Toolbox

To facilitate symmetry annotation, we create a toolbox,[2] as shown in Fig. 3(a). With the toolbox, annotators can simultaneously load images and masks, revise the masks, and save the ground-truth symmetry. The toolbox loads the instance masks and uses a skeleton generation algorithm [27]

[1]Although symmetry is equal to the skeleton in most cases, it is different from the skeleton when the objects have small length-width ratio or occlusion [see Fig. 4(c)] and, thus, is a more accurate word than a skeleton in this article.

[2]https://github.com/KevinKecc/SRN

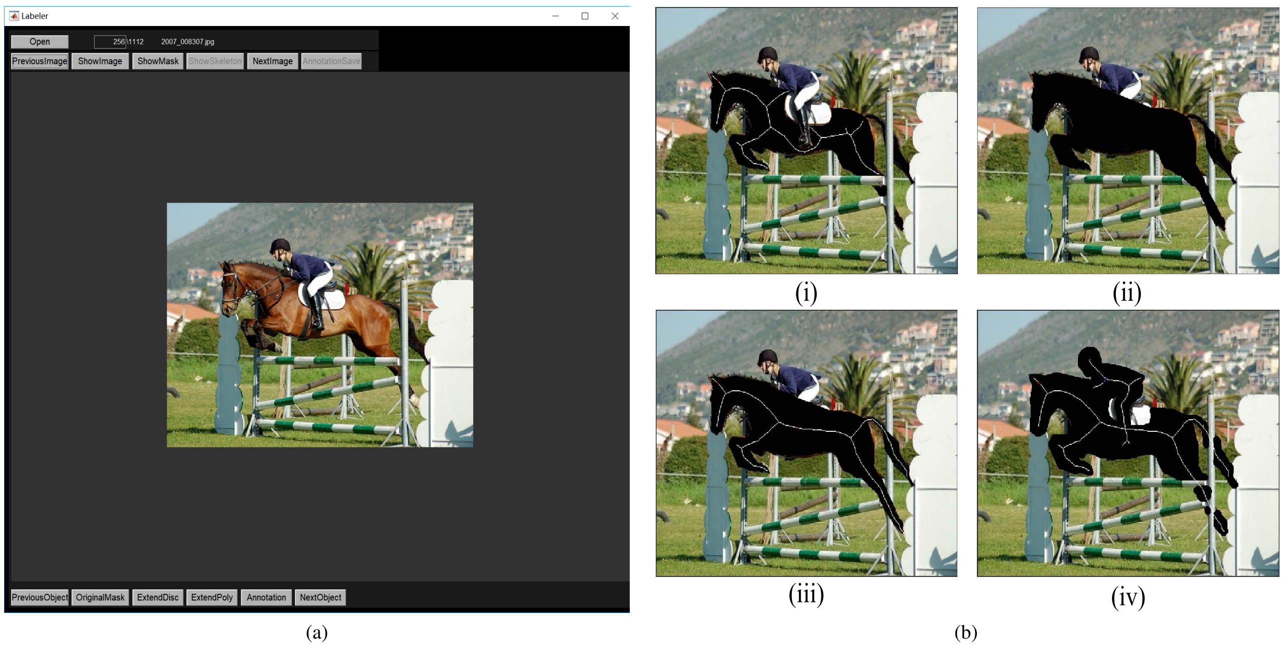


Fig. 3. Reflection symmetry annotation. (a) Annotation Toolbox. Its most salient aspect is the interactive function by which annotators can leverage instance semantic segmentation and skeleton generation algorithms to speed up symmetry annotation. (b) Annotation procedure: (i) Original mask and skeleton generation. (ii) Manual mask extension. (iii) Reflection symmetry extraction for a single object. (iv) Reflection symmetry extraction for multiple objects.

for symmetry annotation. Following the definition of reflection symmetry, annotators can revise the object skeleton to reflection symmetry so that each pixel on the segmentation mask has only one symmetric point subject to the symmetry axis. With the initialization of the object skeleton, an annotator can complete the symmetry for an image within one minute.

Annotation instruction is illustrated in Fig. 3(b). First, it is required to load an image and its instance segmentation mask, as shown in Fig. 3(b)–(i). Second, it is required to use the "Disc" or "Poly" operations defined in the toolbox to obtain an extended mask, as shown in Fig. 3(b)–(ii). The skeletonization is applied to the extended mask to get initial reflection symmetry, as shown in Fig. 3(b)–(iii). Finally, it is required to revise the skeleton to symmetry by cutting the extended skeleton according to the original mask. The instances are annotated one by one, and the ground-truth symmetry of the whole image is completed, as shown in Fig. 3(b)–(iv).

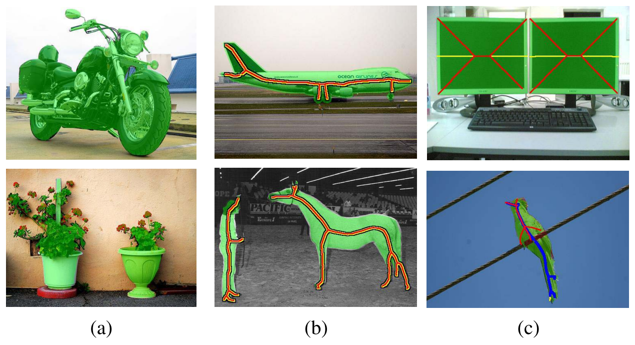


Fig. 4. Comparison of skeleton and object reflection symmetry. The green masks are annotated semantic segmentation ground truth. The red lines are the skeletons of semantic segmentation masks. The yellow lines are the annotated symmetry ground truth. (a) Symmetry unavailable images. (b) Images in which symmetry is equal to the skeleton and is easy to be annotated. (c) Images in which symmetry is hard to be annotated. (Best viewed in color.)

B. Categorization

We annotate the segmentation subset of PASCAL VOC 2011 [17] and release it as Sym-PASCAL. As potted plants, dining tables, motorbikes, bicycles, chairs, and sofas in PASCAL VOC contain a lot of discontinuous parts, the symmetry cannot be annotated, Fig. 4(a). The other 14 object categories in PASCAL VOC are symmetry-available, which is divided into easy ones consisted of slender parts [see Fig. 4(b)] and hard ones with occlusion or small length-width ratio [see Fig. 4(c)]. We name the new symmetry data set as Sym-PASCAL, which is consisted of 648 images for training and 787 images for the test. Among these images, 31.3% are multiobject, i.e., more than one instance in a single image, and 45.6% are part invisibility, i.e., an instance or a part of an instance is occluded by itself and/or other instances, or out the scope of the image.

For the easy annotated images, the skeleton algorithm [27] is utilized directly on the instance mask of the object. As shown in Fig. 4(b), the skeleton is equal to symmetry ground truth for easy annotated images. The hard annotated images are processed with different strategies with wide object and occlusion. As shown on the top of Fig. 4(c), the symmetry for the object with a small long-width ratio, i.e., monitor, is generated by extending the instance mask along the long axis, producing skeleton on the extended mask, and cut the skeleton with original instance mask. As shown at the bottom of Fig. 4(c), the symmetry for the object with occlusion, i.e., bird, is generated by empirically filling the mask of occlusion parts, producing skeleton on the filled mask. One can see that the symmetry ground truth (colored with yellow) is totally different from the skeleton (colored with red), and

TABLE I
COMPARISON OF SKELETON AND SYMMETRY DETECTION DATA SETS

	Data type	Image type	#object	#train -ing	#test -ing
SYMMAX	local symmetry	natural image	–	200	100
WH-SYMMAX	object skeleton	sample image	1	228	100
SK506	object skeleton	sample image	16	300	206
SK-LARGE	object skeleton	sample image	16	746	745
Sym-PASCAL	object symmetry	natural image	14	648	787

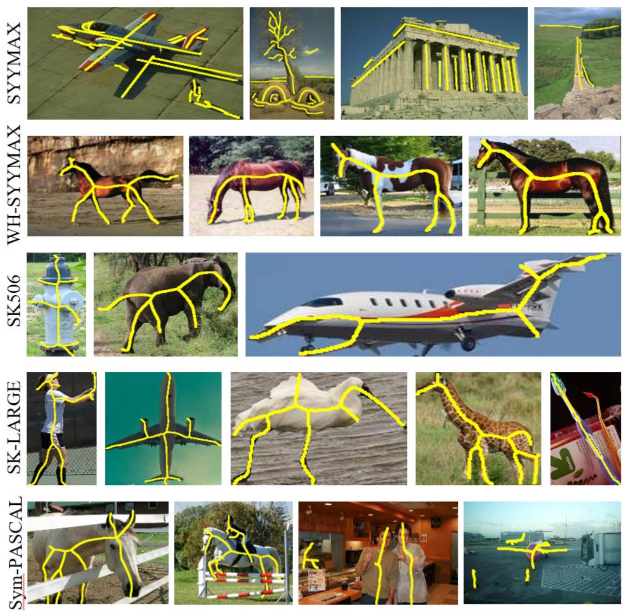


Fig. 5. Symmetry examples from the new released benchmark, Sym-PASCAL, and other four commonly used benchmarks. Compared with the existing data sets, Sym-PASCAL spans challenges, including object diversity, multiobjects, part invisibility, and various complex background.

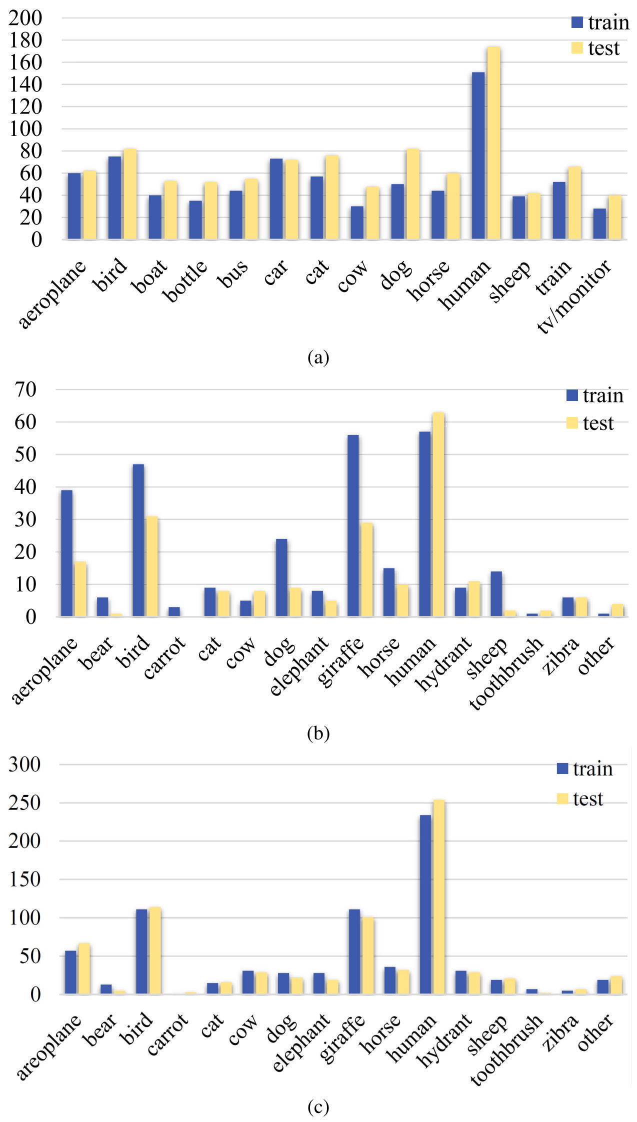


Fig. 6. Object-class distributions of the Sym-PASCAL, SK506, and SK-LARGE data sets. (a) Sym-PASCAL. (b) SK506. (c) SK-LARGE.

the symmetry is more representative than the skeleton for these images [see Fig. 4(c)].

C. Data set Comparison

SYMMAX contains 200 training images and 100 test images, which is annotated on BSDS300 [41]. Both background and foreground are annotated with local reflection symmetry. As foreground usually takes more important position than background [4], [42], it is more meaningful to compute the symmetry for objects (i.e., foreground) instead of the symmetry for the whole image. WH-SYMMAX has 228 training images and 100 test images. It is developed for object skeletons, but it is made up of only cropped horse patches that are not comprehensive for general object symmetry. SK506 involves skeletons from about 16 classes of objects with 300/206 training and test images. SK-LARGE is extended from SK506 to 746/745 training and test images.

Fig. 5 shows some examples, Table I gives the statistic information, and Fig. 6 shows the object-class distribution of different symmetry data sets. Comparing with the existing data sets, the proposed Sym-PASCAL has more images for training and test (see Table I). Particularly, these images involve multiple objects, clustered backgrounds, and/or occlusions (see Fig. 5). In Sym-PASCAL, the number distribution of images for object class is more balanced than other data sets [see Fig. 6(a)]. In contrast, the object distributions are unbalanced in SK506 and SK-LARGE [see Fig. 6(b) and (c)].

IV. SIDE-OUTPUT RESIDUAL NETWORK

SRN is based on the simple yet effective output RU with end-to-end learning.

A. Network Architecture

The SRN is a fully convolutional neural network which stacks RUs on the side outputs of different convolutional stages. By choosing the input of the first stacked RU as the deepest or the shallowest side output, the deep-to-shallow and shallow-to-deep architectures are constructed, as shown in Fig. 7(a) and (b). SRN incorporates the advantages of both

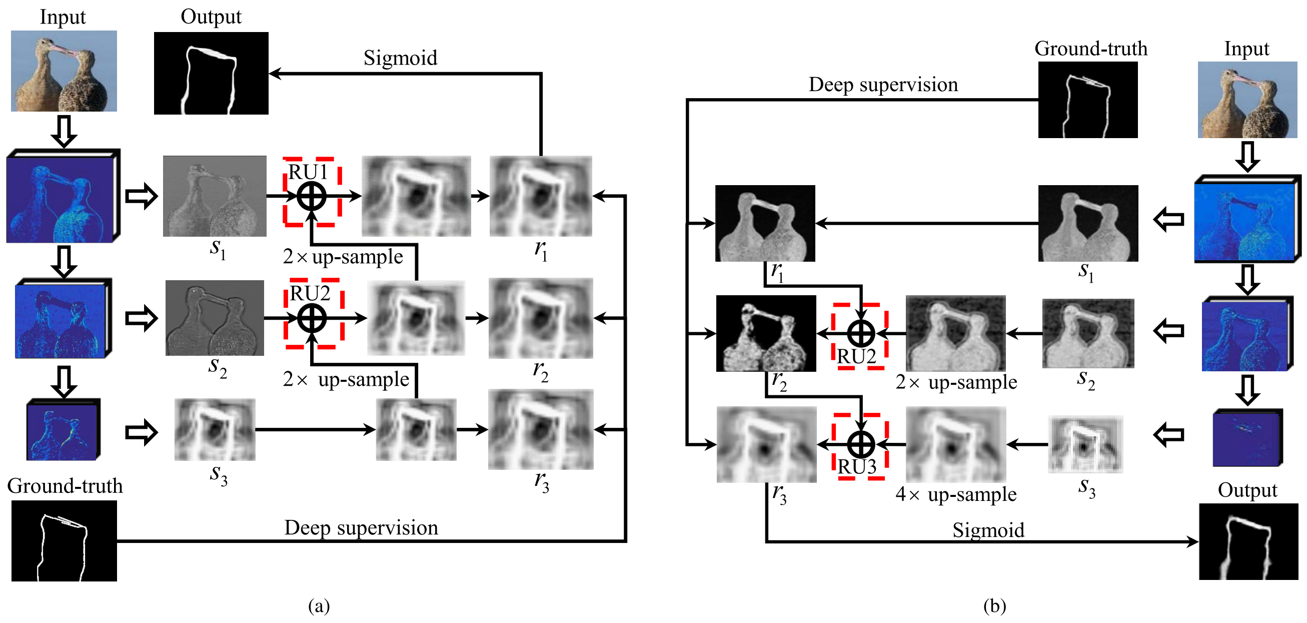


Fig. 7. Architectures of proposed SRN by cascading RUs in (a) deep-to-shallow and (b) shallow-to-deep strategies. The RUs are marked with dashed boxes. With the deep supervision both on the input and output of RUi, $i = 1, 2, 3$, the residual between the ground truth and the output of RUi (i.e., r_i) is computed hierarchically. Along the cascading orientation, the residual decreases so that the RUi (i.e., r_i) is closer to ground truth than the input of RUi (i.e., r_{i+1} in deep-to-shallow, r_{i-1} in shallow-to-deep).

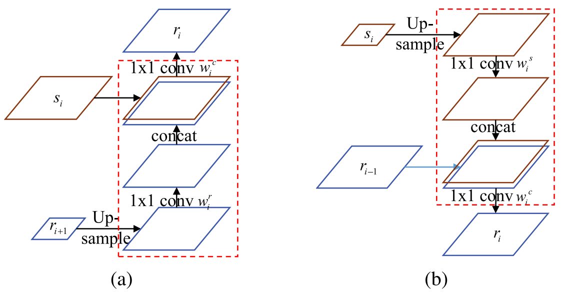


Fig. 8. Implementation of the ith RU. (a) Deep-to-shallow. (b) Shallow-to-deep.

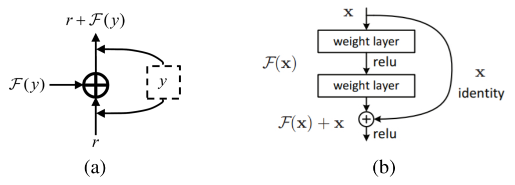


Fig. 9. Output RU (a) in SRN and Residal block (b) in ResNet [43]. By supervision both on the input and output of RU, the additional mapping $\mathcal{F}(y)$ estimates the residual of ground truth y.

residual learning and scale adaptability. The deep-to-shallow architecture is introduced in detail below, and the shallow-to-deep one has a similar implementation.

The implementation of RU in deep-to-shallow is shown in Fig. 8(a). RU$_i$ is denoted as the RU on the side output s_i. The input and output of RU$_i$ are r_{i+1} and r_i, respectively. The side output s_i is utilized to learn the residual between r_{i+1} and the ground truth and updates r_{i+1} to r_i. For the deepest RU, the input is set as the side output, i.e., $r_3 = s_3$ [see Fig. 8(a)]. Note that the size of output r_i is same to the side output s_i rather than the size of input r_{i+1}. Therefore, a Gaussian deconvolutional layer [19] is introduced to upsample r_{i+1}. In this architecture, only $2\times$ upsampling is used as the additional mapping, i.e., $\mathcal{F}(y)$ is $2\times$ larger of the input of RU. A weight layer w_i^r is used to improve the adaptability of scales as the upsampling with the fixed Gaussian kernel is a linear transformation. Instead of hard-add upsampled r_{i+1} and s_i directly, a 1×1 convolutional layer w_i^c is utilized to generate r_i. The ith RU is formulated as

$$r_i = {w_i}^c \otimes \left(s_i + w_i^r \cdot r_{i+1}\right). \tag{3}$$

The convolution operation in 3 can be written as

$$r_i = {w_i}_1^c \cdot s_i + {w_i}_2^c \cdot w_i^r \cdot r_{i+1} \tag{4}$$

where ${w_i}^c$ and ${w_i}^r$ are the convolutional weights of the concatenation layer and the upsampled r_{i+1}. ${w_i}^r$ is a scaler to weight r_{i+1} and ${w_i}^c$ has two elements: one for s_i and the other for r_{i+1}. The output residual between r_i and r_{i+1} is computed as

$$\mathcal{F}_i(y) = {w_i}_1^c \cdot s_i + \left({w_i}_2^c \cdot w_i^r - 1\right) r_{i+1}. \tag{5}$$

When ${w_i}_2^c \cdot w_i^r$ approximates 1.0, the residual is related to only the side output. To the extreme, along the stacking orientation of RUs, the residual $\mathcal{F}(y)$ approximates 0.0.

B. Output Residual Learning Mechanism

The RU for the side output of the trunk network is different from the short-cut connection for feature learning in residual network [43]. The former one approximates a progressive estimation function for ground truth, while the later one eases the training of deep networks. As shown in Fig. 9(b), the conventional residual block uses a shortcut connection between the input x and the output $\mathcal{F}(x) + x$, with the aim to push the residual $\mathcal{F}(x)$ toward 0, while, in Fig. 9(a), our RU involves deep supervision on side outputs r and $r+\mathcal{F}(y)$ and targets at precisely fitting the side-output signal.

The mechanism of RU is shown in Fig. 9(a). Both the input and output of RU are deeply supervised with ground truth. It is forced to learn the residual between the input of RU and the ground truth. Simplify the notification as r, y, and $\mathcal{F}(y)$ that stand for the input of RU, ground truth, and additional mapping, respectively. The RU is formulated as

$$\begin{cases} r \approx y \\ r+\mathcal{F}(y) \approx y \end{cases} \tag{6}$$

where $\mathcal{F}(y)$ is used to estimate the residual of y. By stacking RU on the side output of different stages of the trunk network, it implies a functional module for the "flow" of errors along the stacking direction. It would be easier to push $\mathcal{F}(y)$ to zero than to fit it to ground truth when an input r is optimal.

C. Formulation

Suppose $S = \{(X_n, Y_n)\}_{n=1}^{N}$ is object reflection symmetry training data set with N samples. $X_n = \{x_j^{(n)}, j = 1, \ldots, T\}$ is an input image with T pixels. $Y_n = \{y_j^{(n)}, j = 1, \ldots, T\}$ is the ground-truth symmetry map, where $y_j^{(n)} = 1$ indicates the pixel on the symmetry curve and $y_j^{(n)} = 0$ indicates background. n is drooped for notational simplicity below. Suppose the trunk network with M stages and it produces M side outputs. The SRN stacks $M-1$ RUs and uses the Mth side output as the basic output (the input of first stacked RU). For the basic output, the loss is computed with weighted entropy loss [18], as

$$\begin{aligned} \mathcal{L}_b(\mathbf{W}, w_b) = &-\beta \sum_{j\in Y_+} \log \Pr(y_j = 1|X; \mathbf{W}, w_b) \\ &- (1-\beta) \sum_{j\in Y_-} \log \Pr(y_j = 0|X; \mathbf{W}, w_b) \end{aligned} \tag{7}$$

where Y_+ and Y_- denote the ground-truth label sets indicating a pixel on symmetry curve or background, respectively. $\mathbf{W}$ is the parameters of trunk network. w_b is the parameter for 1×1 convolutional layer, which is used as a pixel-based classifier for the basic output. The loss weight $\beta = |Y_+|/|Y|$, in which $|Y_+|$ and $|Y_-|$ denote the numbers of pixel on symmetry curve and background, respectively. $\Pr(y_j = 1|X; \mathbf{W}, w_b) \in [0, 1]$ is the probability that measures how confident a pixel is on the axis of the symmetry. For the ith RU, where $i = M-1, \ldots, 1$, the loss is computed, as

$$\begin{aligned} \mathcal{L}_i(\mathbf{W}, \theta_i, w_i) = &-\beta \sum_{j\in Y_+} \log \Pr(y_j = 1|X; \mathbf{W}, \theta_i, w_i) \\ &- (1-\beta) \sum_{j\in Y_-} \log \Pr(y_j = 0|X; \mathbf{W}, \theta_i, w_i) \end{aligned} \tag{8}$$

where $\theta_i = (w_i^c, w_i^s)$ is parameter for the ith RU, in which w_i^c and w_i^s are for the convolutional layers after the concatenation layer and side-output layer. w_i is the classifier parameter for the ith side output. The fused loss function is computed by

$$\mathcal{L}(\mathbf{W}, \theta, w) = \mathcal{L}_b(\mathbf{W}, w_b) + \sum_{i=M-1}^{1} \mathcal{L}_i(\mathbf{W}, \theta_i, w_i). \tag{9}$$

We learn the parameters for the trunk network and RUs by

$$(\mathbf{W}^*, \theta^*, w^*) = \arg\min \mathcal{L}(\mathbf{W}, \theta, w). \tag{10}$$

For inference, the output by the last stacked RU is used as symmetry prediction map for the input image X, as

$$\hat{Y} = \Pr(y_j = 1|X; \mathbf{W}^*, \theta^*, w^*). \tag{11}$$

D. Understanding SRN

The proposed SRN leverages output RUs to fit the errors between the object symmetry ground truth and the inputs of RUs. By cascading RUs in a deep-to-shallow manner, SRN approximates a progressive estimation function, which has been successfully used in mechanics and control systems [44] and validated to be more effective than direct linear error estimation. With such progressive estimation, SRN exploits the "flow" of errors among multiple scales to ease the problems of fitting complex outputs with limited convolutional layers. The residual monotonically decreases, and the features are weighted and assembled during learning.

Given M RUs to be stacked, (6) is reformulated as

$$r_o = r_M + \sum_{i=M}^{1} \mathcal{F}_i(y) \tag{12}$$

where r_o is the final output of SRN and $\{r_i\}_{i=1}^{M} \in [0, v_{gt}]$ are the inputs of RUs. $v_{gt} \in \{0, 1\}$ is the ground truth. Mth RU, i.e., r_M, is supervised by ground truth and is seen as an initial output. After the sigmoid operation, the residual keeps positive, i.e., $\mathcal{F}_i(y)_{i=1}^{M} \geq 0$. For the pixels on symmetry axis, we have $1 = v_{gt_+} \geq r_o \geq r_1 \geq \ldots \geq r_M$, which means that the residual monotonically decreases in the order of the stacking RUs, while, for the pixels on background, the network forces $r_o \to 0$.

From the perspective of learning, SRN implements a special kind of learner ensemble, where each learner corresponds to a convolutional layer and can be regarded as a weak learner. Multiple weak learners are assembled to construct a strong learner. The deep-to-shallow SRN first chooses the best weak learner, which uses features from the deepest convolutional layer to fit the ground truth. In the next layer, the error of each sample is calculated, and the samples of larger error contribute more to the second weak learner. Such a procedure could be regarded as a special kind of resampling by assigning larger weights to samples with larger errors. With weak learner ensemble and resampling, the learning procedure of SRN is a boosting-like algorithm [45], and the prediction is a assemble of multiple layer features with loss defined as 9. In the following layers, pixels of larger loss will be assigned with bigger weights.

V. Multitask SRN

SRN consists of a trunk network for feature learning and a branch network, i.e., stacked RUs, for hierarchically side-output processing. When multiple branch networks are added to the trunk, the SRN is upgraded to a MT-SRN (see Figs. 10 and 11). Representation, the buffer layers are used for task-specific feature representation, and stacked RUs are used for output prediction. By combining multiple tasks, we can use

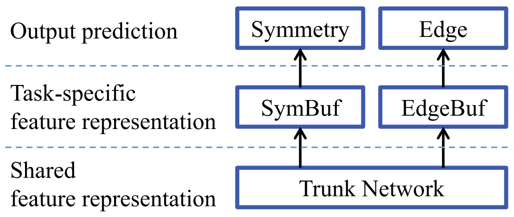

Output prediction
Symmetry
Edge
Task-specific feature representation
SymBuf
EdgeBuf
Shared feature representation
Trunk Network

Fig. 10. General framework of the proposed MT-SRN.

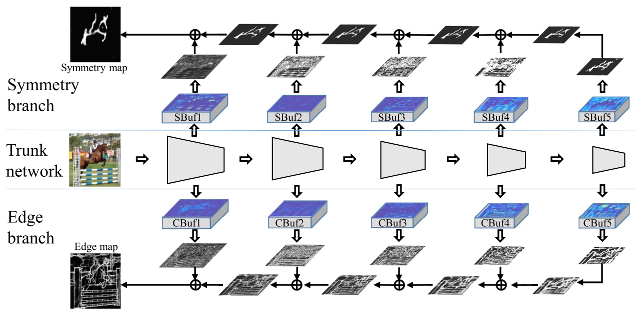


Fig. 11. Architecture of MT-SRN. Buffer convolutional layers (CBuf and SBuf) are added to the trunk network to construct a symmetry branch network and an edge branch network. The MT-SRN is optimized with an alternating training strategy.

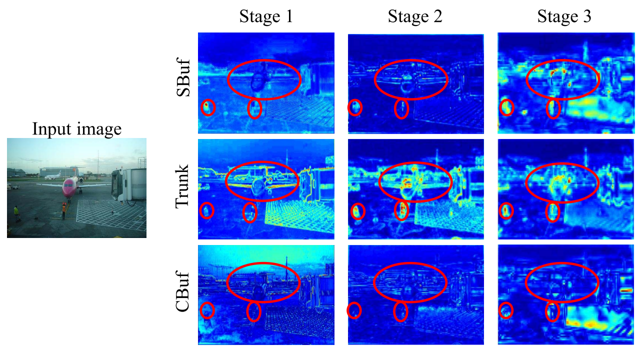


Fig. 12. Illustration of the activated pixels in MT-SRN. The trunk network tends to activate the object, and the Sbuf and Cbuf layers tend to activate edge pixel and symmetry regions, respectively. (Best viewed in color and zoomed in.)

more data to train the shared trunk network to activate object regions and depress backgrounds. MT-SRN is used to perform joint symmetry and edge detection by using a color image as input and predicting the mask as the output of the last stacked RU.

MT-SRN has two loss functions, defined as

$$\mathcal{L}_S = \mathcal{L}(\mathbf{W}_T, \theta_S, w_S), \tag{13}$$

$$\mathcal{L}_E = \mathcal{L}(\mathbf{W}_T, \theta_E, w_E) \tag{14}$$

where $\mathbf{W}_T$ is the parameter of the trunk network, θ_S and θ_E are the parameters of symmetry branch and edge branch, and w_S and w_E are classifiers of the side outputs, respectively. To optimize the two loss functions defined, we propose an alternating training strategy to optimize the following objectives:

$$\left(\mathbf{W}_T^*, \theta_S^*, w_S^*\right) = \arg\min \mathcal{L}(\mathbf{W}_T, \theta_S, w_S), \tag{15}$$

$$\left(\mathbf{W}_T^*, \theta_E^*, w_E^*\right) = \arg\min \mathcal{L}(\mathbf{W}_T, \theta_E, w_E). \tag{16}$$

In the training phase, we first fine-tune the trunk network with the edge branch [see (15)] and then fine-tune the trunk network with the symmetry branch [see (16)]. Five training steps are used in total. The first step is with 8000 iterations, and the number of iterations is reduced by 2000 for each step.

In the MT-SRN, the trunk network is shared for the two tasks that have different learning objectives. In the learning procedure, the network parameters could change drastically when switching between the loss functions defined in (13) and (14). It could aggregate the risk of leaning instability. To solve this problem, we propose to insert a buffer convolutional layer before the side-output layer (see Fig. 9). Buffer layers have been successfully used in a multiscale region proposal network (RPN) [46] to alleviate the problem of sharp gradient values.

In MT-SRN, the buffer layers are introduced to smooth the gradient values propagated from either branch network and guarantee that the parameters of the trunk network update smoothly. Besides, the buffer layer is used to extract the task-specific feature. As shown in Fig. 12, the trunk network tends to activate the object regions, and the Sbuf and Cbuf layers tend to activate edge pixel and symmetry regions, respectively. This inspires the interpretation of SRN by demonstrating that high-level semantics can be abstracted from low-level features with the trunk network, while object-oriented low-level features (symmetry and edge) can be extracted by fusing low-level features with high-level semantics using SRN.

We can conclude the advantages of MT-SRN: 1) In the training phase, the shared trunk network of MT-SRN is fine-tuned by multiple data sets, e.g., the edge and symmetry data sets. This aggregates the limited training data from each data set; and 2) in the test phase, MT-SRN uses a shared trunk network and forward the trunk network once to perform two detection tasks. It saves both storage space and computational time comparing to two separated SRNs. The more tasks it performs, the more storage space and computational time it saves.

VI. Experiments

In this section, the implementation of SRN for symmetry detection is first presented. The experimental results of symmetry detection are then analyzed and compared. Finally, the efficiency and effectiveness of MT-SRN for joint symmetry and edge detection are validated.

SRN is built on the publicly available implementation of the deep network for holistically nested edge detection, HED [18]. The final output map of SRN is postprocessed with a nonmaximal suppression (NMS) algorithm [47], generating one-pixel width object symmetry or edge curves. The result difference between SRN and HED is illustrated in Fig. 13. The final output of HED is the average of weighted side outputs, while that of SRN is achieved by cascading the side outputs hierarchically from the deep to the shallow stages. As shown in the first and third rows of Fig. 13, the side outputs from shallow stages of HED are about local structures, and those from deep stages are about global contours. The weighted summing of multistage side outputs in HED could introduce local noises to the final output. To alleviate the noise and preserve the global contours, the SRN stacks the multistage side outputs, with which the residuals between the RU-outputs

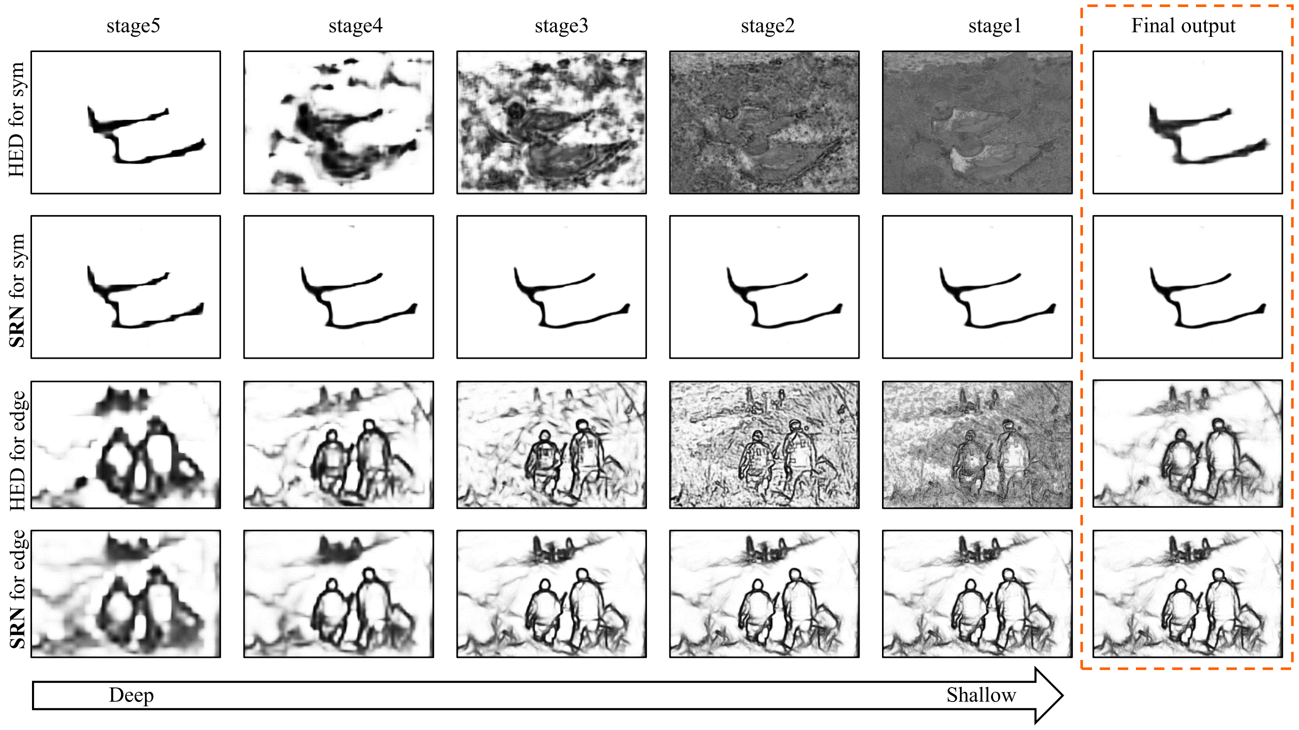


Fig. 13. Comparison of the side outputs of HED [18] and single SRNs. From the first to the fifth column, we illustrate the side outputs from deep to shallow stages. The last column is the final output. For SRN (the second and fourth rows), it can be seen that the residuals between the outputs and the ground truth decrease progressively. In contrast, HED (the first and third rows) has no such characteristic.

TABLE II
PERFORMANCE OF SRN UNDER DIFFERENT IMPLEMENTATIONS ON THE SYM-PASCAL BENCHMARK

Architecture	Augumentation	Conv1	F-measure
shallow-deep	$1\times$	with	0.381
		w/o	0.397
	$0.8\times, 1\times, 1.2\times$	with	0.371
		w/o	0.396
deep-shallow	$1\times$	with	**0.443**
		w/o	**0.443**
	$0.8\times, 1\times, 1.2\times$	with	0.384
		w/o	0.397

and the ground truth decrease progressively, as shown in the second and fourth rows of Fig. 13. This validates the monotonicity of residual in the order of the stacking RUs, as defined in 12.

A. SRN Implementation

We use F-measure and precision–recall curve for performance evaluation following [13].

Parameters: For both object reflection symmetry detection and edge detection, we set the minibatch size to 1, the momentum to 0.9, the weight decay to 0.002, the maximum number of training iterations to 20 000, and the learning rate to 1e-8 for in-the-wild image data sets and 1e-6 for simple image data sets.

Architecture: Table II shows that SRN with the deep-to-shallow architecture (F-measure 0.443) performs significantly better than the shallow-to-deep architecture (F-measure 0.397) on Sym-PASCAL data set. It confirms that the deep-to-shallow architecture is easier to reduce the residual than the shallow-to-deep one as its initialization is better, as analyzed in Section IV-A.

Data Augmentation: Data augmentation can aggregate the training data sets. In this work, image rotation, flipping, upsampling, and downsampling (multiscale) are used for data argumentation, following [15]. For each scale, we rotate the training images every 90° and flip them along a different axis. The performance with/without multiscale data argumentation is compared. Table II shows that the F-measure decreases when using multiscale augmentation even though it produces more training data. The reason is that the ground-truth symmetry is made up of curves with one-pixel thickness. The upsampling operation produces curves that have thickness lager than one pixel, and the down-sampling operation produces discontinuous symmetry curves.

Conv1: SRN is trained without the conv1 stage of VGG because the size of the receptive field is so small (only 5×5) that it introduces local noise of symmetry (too small to capture any symmetry response). The negative impact of a small receptive field with FSDS [15] is also observed. By pairwise comparison in Table II, the F-measure without conv1 is slightly better than that with conv1.

B. Symmetry Detection

The proposed SRN is evaluated on both Sym-PASCAL and the other four data sets, *i.e.*, SYMMAX, WH-SYMMAX, SK506, and SK-LARGE. The different characteristics of these data sets have been compared in Section III.

1) Performance on Sym-PASCAL: The PR-curve and F-measure of SRN and state-of-the-art are shown in Fig. 14 and Table III. We ran the source code with default parameters

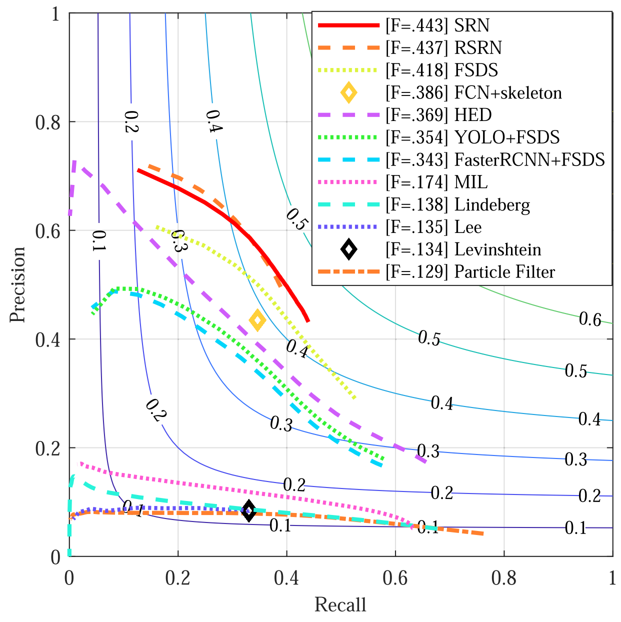


Fig. 14. Performance comparison on the Sym-PASCAL data set.

TABLE III

PERFORMANCE COMPARISON ON THE SYM-PASCAL DATA SET BETWEEN SRN AND THE STATE-OF-THE-ART APPROACHES. †GPU TIME WITH NVIDIA TESLA K40

Approaches		F-measure	Runtime/s
Conventional methods	Particle Filter [24]	0.129	25.3
	Levinshtein [22]	0.134	183.87
	Lee [23]	0.135	685.94
	Lindeberg [49]	0.138	5.79
	MIL [13]	0.174	80.35
Two-stage deep learning methods	Faster-RCNN [50] + FSDS [15]	0.343	0.33 †
	YOLO [51] + FSDS [15]	0.354	0.12 †
	FCN [19] + skeleton [27]	0.386	0.76 †
End-to-end deep learning methods	HED (baseline) [18]	0.369	**0.10** †
	FSDS [15]	0.418	0.12 †
	SRN (ours)	**0.443**	0.12 †
	RSRN (ours)	**0.437**	0.13 †

to achieve the performance of the existing approaches. We also compared the result of Rich-SRN (RSRN) [51], in which we used all the 13 convolutional layers of VGG.

In Table III, one can find that the conventional approaches have poor performance with long running time. MIL [13] achieves the best F-measure among these conventional approaches. The F-measure is 0.174, reflecting that the released Sym-PASCAL is challenging. The fastest conventional approach, i.e., Lindeberg [48], takes 5.79 s per image. Other approaches take more time for running as they use complex features comparing with [48].

The end-to-end deep learning methods perform significantly better than the conventional ones, both on the computational efficiency and detection performance (see Table III). The F-measures of traditional approaches are less than 0.18, while they are larger than 0.34 for deep learning-based methods. It is about 2x better. It takes 5.79 s at least to extract object reflection symmetry for one image with conventional approaches, while it takes 0.76 s at most

TABLE IV

PERFORMANCE COMPARISON OF THE STATE-OF-THE-ART APPROACHES ON THE SYMMAX, WH-SYMMAX, SK506, AND SK-LARGE DATA SETS

datasets	SYMMAX	WH-SYMMAX	SK506	SK-LARGE
Levinshtein [22]	–	0.174	0.217	0.243
Lee [23]	–	0.223	0.252	0.255
Lindeberg [49]	0.360	0.277	0.227	0.270
Particle Filter [24]	–	0.334	0.226	–
MIL [13]	0.362	0.365	0.392	0.293
HED [18]	0.427	0.732	0.542	0.586
FSDS [15]	**0.467**	0.769	0.623	0.633
Seg-Skel [52]	–	–	–	**0.730**
SRN(ours)	0.446	0.780	**0.632**	0.678
RSRN(ours)	0.462	**0.792**	0.622	0.685

for deep learning-based methods. It is about 8x faster. Specifically, HED gets F-measure with 0.369 and takes only ten milliseconds to process an image. FSDS reaches F-measure with 0.418 by concatenating and slicing of all side outputs. The proposed SRN and RSRN achieve the highest F-measure 0.443 and 0.437, which, respectively, outperforms the baseline HED approach by 7.4% and 6.8% and the state-of-the-art FSDS approach by 2.5% and 1.9%.

We give some reflection symmetry detection samples in Fig. 15. From the first to the last column, it illustrates one-object images without clustered background, one-object images with clustered background, multiobject images without clustered background, multiobject images with clustered background, one-object images with occluded objects, and multiobject images with occluded objects, respectively. The proposed SRN is more consistent with the ground truth with/without clustered background and achieves more accurate results for multiobject and occlusion than other approaches.

To show the effectiveness of the end-to-end framework for clustered backgrounds, we compare the proposed SRN with two-stage approaches, which extracts symmetry on the cropped object patch with object detection or masks from instance semantic segmentation. The state-of-the-art segmentation network FCN-8s [19] is used to localize objects in the first stage, and the skeleton method [27] is used to extract symmetry in the second stage. It obtains an F-measure of 0.386, as shown in Fig. 14. We also compare the FSDS [15] on the detection results from the state-of-the-art object detection methods: Faster-RCNN [49] and YOLO [50]. In these two approaches, the FSDS trained on SK506 is performed on object regions produced by Faster-RCNN or YOLO. As shown in Fig. 14, the F-measures are 0.343 and 0.354, respectively. Comparing with the end-to-end SRN, two-stage approaches introduce error accumulation from the semantic segmentation and the object detection steps. Besides, semantic segmentation cannot obtain homogeneous masks when an occlusion exists.

2) Performance on Other Data Sets: The F-measures on the other four symmetry data sets are shown in Table IV. Similar to Sym-PASCAL, the deep learning-based methods achieve significantly better performance on all the data sets than the conventional methods, especially for the simple image data sets, WH-SYMMAX, SK506, and SK-LARGE. The proposed SRN improves the baseline HED with F-measure from 0.427 to 0.446, 0.732 to 0.780, 0.542 to 0.632, and

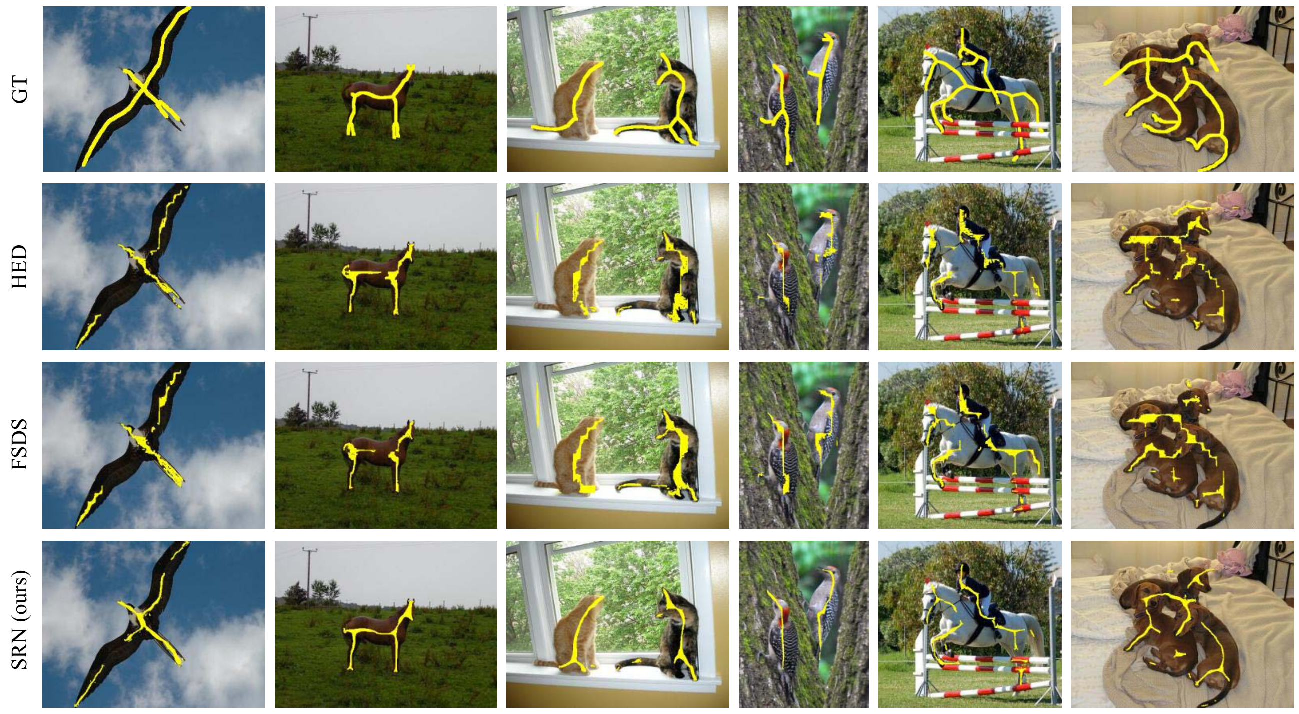


Fig. 15. Object reflection symmetry detection results on the Sym-PASCAL data set. From first to last column, it illustrates one-object images without clustered background, one-object images with clustered background, multiobject images without clustered background, multiobject images with clustered background, one-object images with occluded objects, and multiobject images with occluded objects, respectively.

0.586 to 0.678 on SYMMAX, WH-SYMMAX, SK506, and SK-LARGE, respectively. RSRN further achieves 1.2% and 0.7% performance gain on WH-SYMMAX and SK-LARGE.

In Table IV, RSRN outperforms SRN on all data sets except the SK506. RSRN outperforms SRN on all data sets except the SK506. As all the side outputs of the convolutional layer in VGG-16 are used to construct stack RUs in RSRN, it has richer representation (more parameters) than SRN. Nevertheless, with more parameters to learn, RSRN is easier to overfit a small data set, e.g., SK506.

On WH-SYMMAX and SK506, we achieve the best performance. On SYMMAX, we achieve comparable performance with FSDS. On SK-LARGE, SRN is about 4% lower than Seg-Skel. We note that SegSkel uses semantic segmentation cues and is computationally complex.

C. Edge Detection

The BSDS500 [41] data set is used to evaluate the performance of edge detection, which is composed of 200 training, 100 validation, and 200 test images. Each image is manually annotated by five persons. For training images, we preserve their positive labels annotated by at least three human annotators, following [41]. Edge detection is evaluated with three standard measures [41]: fixed contour threshold (ODS) that is the same as F-measure, per-image best threshold (OIS), and average precision (AP).

1) Performance: After the sigmoid process, the output from the last stacked RU produces an edge mask. In Table V and Fig. 16, the CNN-based methods significantly outperform conventional methods. Specifically, DeepContour [28] has very good performance but is computational expensive. The HED [18] achieves better performance with ODS = 0.780, which is computed fast at 2.5 frames/s. With negligible computational cost overhead, the proposed SRN achieves ODS = 0.782 at 2.3 frames/s, which outperforms the baseline HED method.

TABLE V

EDGE DETECTION PERFORMANCE EVALUATION ON BSDS500

Methods	ODS	OIS	AP	FPS
Human	0.800	0.800	–	–
Canny [53]	0.590	0.620	0.578	15
Sketch Tokens [54]	0.721	0.739	0.768	1
SE [47]	0.739	0.759	0.792	2.5
DeepContour [28]	0.757	0.776	0.790	0.03
HED [18]	0.780	0.797	**0.814**	2.5
SRN (ours)	**0.782**	**0.800**	0.779	2.3

D. Learning Convergence

The learning convergence of HED and SRN on Sym-PASCAL is shown in Fig. 17. It illustrates that HED has a learning convergence problem even though it achieves good performance on symmetry and edge detection tasks. The proposed SRN tends to converge, benefiting from the output residual fitting. HED needs 12 000 learning iterations to achieve the best performance. Benefited from the RUs, SRN needs only 3000 iterations to achieve the same performance.

E. Multitask SRN

In MT-SRN, VGG is also used as the trunk network, and the settings of the buffer layers are the same as those of the connected convolutional layers in the trunk network. The five training steps are run with 8000, 6000, 4000, 2000, and 1000 learning iterations, respectively. In each training step, the edge and symmetry branches are trained alternatively.

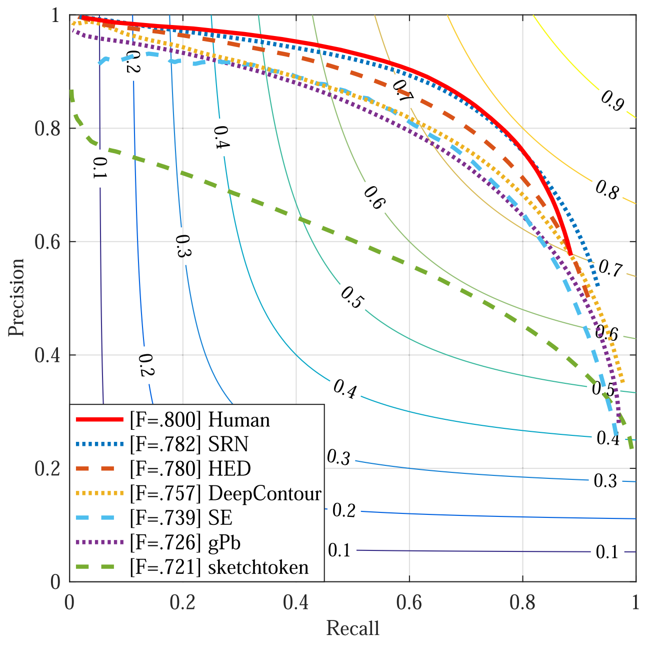


Fig. 16. Edge detection results on BSDS500.

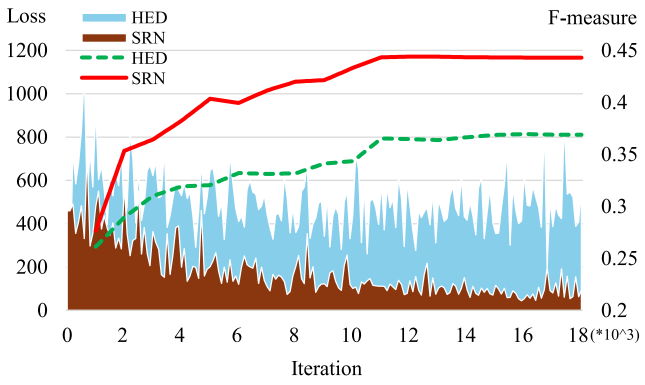
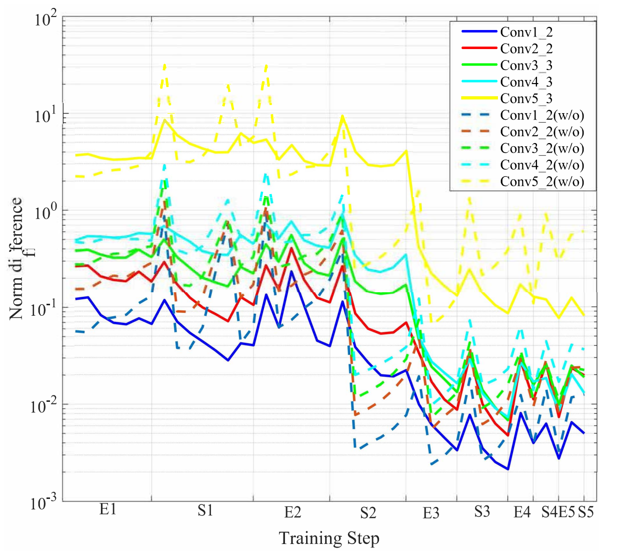
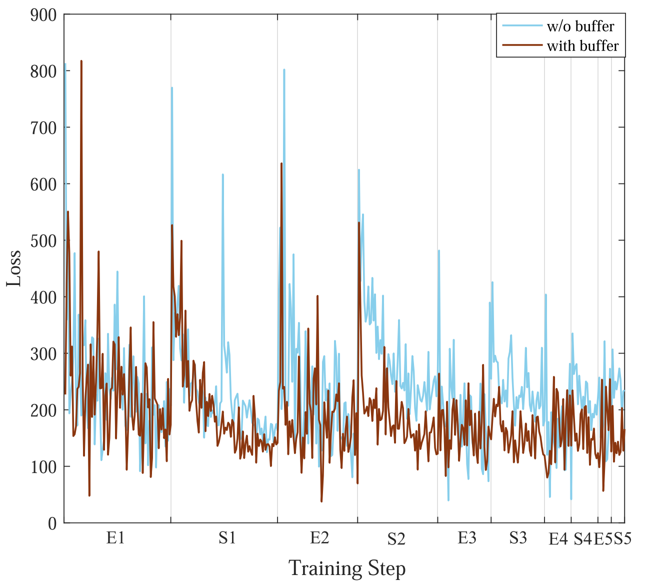


Fig. 17. Loss and F-measure comparison on Sym-PASCAL.



(a)



(b)

Fig. 18. Comparison of (a) norm difference and (b) loss curve of MT-SRN with and without buffer layers.

The learning rate is 1e-6 for the first two steps and 1e-8 for the last three steps.

As discussed in Section V, the stability of the trunk network parameters in MT-SRN is important as it is related to learning convergence. Such stability is measured with the norm difference of the convolutional parameters during the learning iterations. Fig. 18(a) shows that the stability decreases when the training branch switches from one to another. For example, in the first learning stage, when the learning switching from the edge branch to the symmetry branch at the 8000th iteration, the norm difference is large. It is also observed that after the five steps of learning, the difference decreases to a very small value. The decreasing norm difference signifies the increasing of the network stability, which not only validates the effectiveness of buffer layers but also the plausibility of MT-SRN. The norm difference of MT-SRN without the buffer layer is also shown in Fig. 18(a), which oscillates larger than the MT-SRN with the buffer layer. It results in that the loss of MT-SRN without buffer layer is not convergence as good as the MT-SRN with buffer layer [see Fig. 18(b)].

We use the abovementioned data sets for symmetry and edge detection to evaluate the performance of MT-SRN (see Table VI). Surprisingly, the performances for both tasks increase. When BSDS and SYMMAX are used for joint edge and symmetry detection, the ODS of edge detection is improved from 0.782 to 0.785, and the F-measure of symmetry detection improves from 0.446 to 0.464, which updates the state of the art (see Table VI). The other four groups of data sets, i.e., (BSDS500, WH-SYMMAX), (BSDS500, SK506), (BSDS500, SK-LARGE), and (BSDS500, Sym-PASCA), have the similar improvement. There are two reasons for performance improvement. One is that more training iterations (42 000) are used for the trunk network so that MT-SRN is trained better than individual SRN (20 000 iterations). This advantage is observed by comparing it with a specific task with a singular data set in Fig. 17, and the F-measure of SRN does not increase with more iterations. The other is that two different data sets mean more training images so that the diversity of the training set increases. The fact that more training data can improve the performance that is illustrated in object detection

TABLE VI
RESULTS OF MT-SRN FOR EDGE AND SYMMETRY DETECTION TASKS SIMULTANEOUSLY

Method	Datasets	ODS (edge)	F-measure (symmetry)
SRN	BSDS500	0.782	–
	SYMMAX	–	0.446
	WH-SYMMAX	–	0.780
	SK506	–	0.632
	SK-LARGE	–	0.678
	Sym-PASCAL	–	0.443
MT-SRN (w/o)	(BSDS500, SYMMAX)	0.768	0.441
	(BSDS500, WH-SYMMAX)	0.763	0.772
	(BSDS500, SK506)	0.771	0.620
	(BSDS500, SK-LARGE)	0.781	0.652
	(BSDS500, Sym-PASCAL)	0.773	0.436
MT-SRN	(BSDS500, SYMMAX)	0.785	**0.464**
	(BSDS500, WH-SYMMAX)	0.779	**0.807**
	(BSDS500, SK506)	**0.786**	**0.639**
	(BSDS500, SK-LARGE)	**0.786**	**0.681**
	(BSDS500, Sym-PASCAL)	0.784	**0.453**

when multiple data sets for a single task are combined [49]. We further validate that the proposed MT-SRN is feasible to train a shared deep network on data sets from different image-to-mask tasks, which is very useful for tasks where there is no significant training data available. In Table VI, it can be seen that the MT-SRN with buffer layers reports higher performance than MT-SRN without buffer layers for both object reflection symmetry detection and edge detection, as the buffer layer is used as a feature transformation function, and each pixel has task-specific feature representation.

The general applicability presented by MT-SRN is using a shared trunk network and separated branches to perform multitask learning. This helps build deep learning models that can save both storage space and computational cost. Specifically, two separated SRN models need 115 060 KB space, whereas the MT-SRN model needs 78 966 KB. MT-SRN saves 36 094 KB.

VII. CONCLUSION

Object reflection symmetry detection has great applicability in computer vision yet remains unsolved, as indicated by the low performance (often lower than 0.5) of the state-of-the-art methods. In this article, we introduce a new data set for reflection symmetry and propose an end-to-end SRN, establishing a strong baseline for object reflection symmetry detection for natural images. The new data set, with challenges related to real-world images, is validated to be a good touchstone of various state-of-the-art approaches. The proposed SRN, with well-defined and stacked RUs, is validated to be more effective to perform symmetry detection in complex background.

With the adaptability to object scales, the robustness to complex backgrounds, and the end-to-end learning architecture, SRN provides insights into network architecture design. We have extended the SRN to process a similar image-to-mask computer vision task, i.e., edge detection. With multiple network branches designed and alternating training strategy, SRN is upgraded to MT-SRN for joint symmetry and edge detection, showing great potential to process the image-to-mask tasks that are lack of training data. In the future, more effective SRN architectures could be explored by optimizing side-output connects and RUs with emerging network architecture search (NAS) method [55].

ACKNOWLEDGMENT

The authors would like to thank Prof. D. Doermann and Dr. P. Wei for their discussion and suggestions.

REFERENCES

[1] T. B. Sebastian, P. N. Klein, and B. B. Kimia, "Recognition of shapes by editing their shock graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 5, pp. 550–571, May 2004.
[2] N. H. Trinh and B. B. Kimia, "Skeleton search: Category-specific object recognition and segmentation using a skeletal shape model," *Int. J. Comput. Vis.*, vol. 94, no. 2, pp. 215–240, 2011.
[3] C. L. Teo, C. Fermuller, and Y. Aloimonos, "Detection and segmentation of 2D curved reflection symmetric structures," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1644–1652.
[4] H. Fu, X. Cao, Z. Tu, and D. Lin, "Symmetry constraint for foreground extraction," *IEEE Trans. Cybern.*, vol. 44, no. 5, pp. 644–654, May 2014.
[5] T. Lee, S. Fidler, and S. Dickinson, "Learning to combine mid-level cues for object proposal generation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1680–1688.
[6] X. Bai, X. Wang, L. J. Latecki, W. Liu, and Z. Tu, "Active skeleton for non-rigid object detection," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep. 2009, pp. 575–582.
[7] Z. Zhang, W. Shen, C. Yao, and X. Bai, "Symmetry-based text line detection in natural scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 2558–2567.
[8] L. Lam, S.-W. Lee, and C. Y. Suen, "Thinning methodologies—A comprehensive survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 9, pp. 869–885, Sep. 1992.
[9] P. K. Saha, G. Borgefors, and G. S. D. Baja, "A survey on skeletonization algorithms and their applications," *Pattern Recognit. Lett.*, vol. 76, pp. 3–12, Jun. 2016.
[10] J. Liu *et al.*, "Symmetry detection from realworld images competition 2013: Summary and results," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2013, pp. 200–205.
[11] Y. Liu, H. Hel-Or, C. S. Kaplan, and L. J. V. Gool, "Computational symmetry in computer vision and computer graphics," *Found. Trends Comput. Graph. Vis.*, vol. 5, nos. 1–2, pp. 1–195, 2010.
[12] S. Lee and Y. Liu, "Curved glide-reflection symmetry detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 2, pp. 266–278, Feb. 2012.
[13] S. Tsogkas and I. Kokkinos, "Learning-based symmetry detection in natural images," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Oct. 2012, pp. 41–54.
[14] W. Shen, X. Bai, Z. Hu, and Z. Zhang, "Multiple instance subspace learning via partial random projection tree for local reflection symmetry in natural images," *Pattern Recognit.*, vol. 52, pp. 306–316, Apr. 2016.
[15] W. Shen, K. Zhao, Y. Jiang, Y. Wang, Z. Zhang, and X. Bai, "Object skeleton extraction in natural images by fusing scale-associated deep side outputs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 222–230.
[16] W. Shen, K. Zhao, Y. Jiang, Y. Wang, X. Bai, and A. Yuille, "DeepSkeleton: Learning multi-task scale-associated deep side outputs for object skeleton extraction in natural images," *IEEE Trans. Image Process.*, vol. 26, no. 11, pp. 5298–5311, Nov. 2017.
[17] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. (2011). *The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results*. [Online]. Available: http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html
[18] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1395–1403.
[19] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.
[20] W. Ke, J. Chen, J. Jiao, G. Zhao, and Q. Ye, "SRN: Side-output residual network for object symmetry detection in the wild," in *Proc. CVPR*, 2017, pp. 302–310.
[21] X. Bai, L. Latecki, and W.-Y. Liu, "Skeleton pruning by contour partitioning with discrete curve evolution," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 3, pp. 449–462, Mar. 2007.

[22] A. Levinshtein, C. Sminchisescu, and S. Dickinson, "Multiscale symmetric part detection and grouping," *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 117–134, Sep. 2013.

[23] T. S. H. Lee, S. Fidler, and S. Dickinson, "Detecting curved symmetric parts using a deformable disc model," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 1753–1760.

[24] N. Widynski, A. Moevus, and M. Mignotte, "Local symmetry detection in natural images using a particle filtering approach," *IEEE Trans. Image Process.*, vol. 23, no. 12, pp. 5309–5322, Dec. 2014.

[25] C. Funk *et al.*, "2017 ICCV challenge: Detecting symmetry in the wild," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 1692–1701.

[26] Z. Yu and C. Bajaj, "A segmentation-free approach for skeletonization of gray-scale images via anisotropic vector diffusion," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2004, pp. 415–420.

[27] W. Shen, X. Bai, R. Hu, H. Wang, and L. J. Latecki, "Skeleton growing and pruning with bending potential ratio," *Pattern Recognit.*, vol. 44, no. 2, pp. 196–209, Feb. 2011.

[28] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang, "DeepContour: A deep convolutional feature learned by positive-sharing loss for contour detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3982–3991.

[29] S. Hallman and C. C. Fowlkes, "Oriented edge forests for boundary detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1732–1740.

[30] J. Yang, B. Price, S. Cohen, H. Lee, and M.-H. Yang, "Object contour detection with a fully convolutional encoder-decoder network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 193–202.

[31] G. Bertasius, J. Shi, and L. Torresani, "High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 504–512.

[32] G. Ghiasi and C. C. Fowlkes, "Laplacian pyramid reconstruction and refinement for semantic segmentation," in *Proc. Europ. Conf. Comput. Vis. (ECCV)*, Oct. 2016, pp. 519–534.

[33] C. Funk and Y. Liu, "Beyond planar symmetry: Modeling human perception of reflection and rotation symmetries in the wild," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 793–803.

[34] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640–651, Apr. 2017.

[35] J. Dai, K. He, and J. Sun, "Instance-aware semantic segmentation via multi-task network cascades," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3150–3158.

[36] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1997.

[37] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *Proc. IEEE Int. Conf. Learn. Represent. (ICLR)*, 2014, pp. 5454–5463.

[38] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 2650–2658.

[39] R. Ranjan, V. M. Patel, and R. Chellappa, "HyperFace: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 1, pp. 121–135, Jan. 2019.

[40] I. Kokkinos, "UberNet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5454–5463.

[41] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, May 2011.

[42] A. Borji, M.-M. Cheng, H. Jiang, and J. Li, "Salient object detection: A benchmark," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 5706–5722, Dec. 2015.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[44] D. De Santis, "High-order linear and non-linear residual distribution schemes for turbulent compressible flows," *Comput. Methods Appl. Mech. Eng.*, vol. 285, pp. 1–31, Mar. 2015.

[45] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. 13th Int. Conf. Mach. Learn. (ICML)*, Jul 1996, pp. 148–156.

[46] Z. Cai, Q. Fan, R. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Oct. 2016, pp. 354–370.

[47] P. Dollar and C. L. Zitnick, "Fast edge detection using structured forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 8, pp. 1558–1570, Aug. 2015.

[48] T. Lindeberg, "Edge detection and ridge detection with automatic scale selection," *Int. J. Comput. Vis*, vol. 30, no. 2, pp. 117–156, 1998.

[49] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2015, pp. 91–99.

[50] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[51] C. Liu, W. Ke, J. Jiao, and Q. Ye, "RSRN: Rich side-output residual network for medial axis detection," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 1739–1743.

[52] X. Liu, P. Lyu, X. Bai, and M.-M. Cheng, "Fusing image and segmentation cues for skeleton extraction in the wild," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 1744–1748.

[53] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. TPAMI-8, no. 6, pp. 679–698, Nov. 1986.

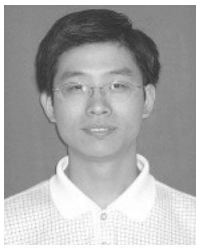
[54] J. J. Lim, C. L. Zitnick, and P. Dollar, "Sketch tokens: A learned mid-level representation for contour and object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 3158–3165.

[55] C. Liu *et al.*, "Progressive neural architecture search," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 19–35.

Wei Ke (Member, IEEE) received the Ph.D. degree in electrical engineering from the University of Chinese Academy of Sciences, Beijing, China, in 2018.

He is currently an Associate Professor with Xi'an Jiaotong University, Xi'an, China. He was a Post-Doctoral Researcher with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA, until 2020. In 2016, he visited the Center for Machine Vision and Signal Analysis, University of Oulu, Oulu, Finland, as a joint Ph.D. student, supported by the China Scholarship Council (CSC). He has authored or coauthored more than 20 articles in refereed conferences and journals including IEEE CVPR and ECCV. His research interests include computer vision and deep learning.

Dr. Ke is the winner of the President Award of Chinese Academy of Sciences in 2017.



Jie Chen (Member, IEEE) received the M.S. and Ph.D. degrees from the Harbin Institute of Technology, Harbin, China, in 2002 and 2007, respectively.

He joined the Faculty with the Graduate School in Shenzhen, Peking University, in 2019, where he is currently an Associate Professor with the School of Electronic and Computer Engineering. Since 2018, he has been working with the Peng Cheng Laboratory, Shenzhen, China. From 2007 to 2018, he worked as a Senior Researcher with the Center for Machine Vision and Signal Analysis, University of Oulu, Oulu, Finland. In 2012 and 2015, he visited the Computer Vision Laboratory, University of Maryland, College Park, MD, USA, and the School of Electrical and Computer Engineering, Duke University, Durham, NC, USA, respectively. His research interests include deep learning, computer vision, and medical image analysis.

Dr. Chen was a Co-Chair of International Workshops at ACCV, CVPR, ICCV, and ECCV. He was a Guest Editor of special issues for IEEE TPAMI, IJCV, and *Neurocomputing*. He is an Associate Editor of *The Visual Computer*.

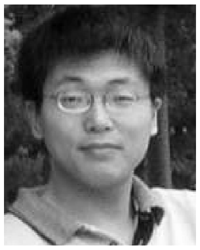
Jianbin Jiao (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in mechanical and electronic engineering from Harbin Institute of Technology (HIT), Harbin, China, in 1989, 1992, and 1995, respectively.

From 1997 to 2005, he was an Associate Professor with HIT. Since 2006, he has been a Professor with the School of Electronic, Electrical, and Communication Engineering, University of the Chinese Academy of Sciences, Beijing, China. His research interests include image processing, pattern recognition, and intelligent surveillance.

Guoying Zhao (Senior Member, IEEE) received the Ph.D. degree in computer science from the Chinese Academy of Sciences, Beijing, China, in 2005.

She is currently a Professor with the Center for Machine Vision and Signal Analysis, University of Oulu, Oulu, Finland, where she has been a Senior Researcher since 2005 and an Associate Professor since 2014. She has authored or coauthored more than 220 articles in journals and conferences. Her articles have currently over 12 160 citations in Google Scholar (h-index 50). Her current research interests include image and video descriptors, facial-expression and micro-expression recognition, gait analysis, dynamic-texture recognition, human motion analysis, and person identification. Her research has been reported by Finnish TV programs, newspapers, and *MIT Technology Review.*

Dr. Zhao was a Co-Publicity Chair for FG2018, a General chair of 3rd International Conference on Biometric Engineering and Applications (ICBEA 2019), and Late Breaking Results Co-Chairs of 21st ACM International Conference on Multimodal Interaction (ICMI 2019), has served as an area chair for several conferences, and is an Associate Editor for *Pattern Recognition*, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, and *Image and Vision Computing Journals*. She has lectured tutorials at ICPR 2006, ICCV 2009, SCIA 2013, and FG 2018, authored/edited three books, and eight special issues in journals. She was a Co-Chair of many International Workshops at ICCV, CVPR, ECCV, ACCV, and BMVC.



Qixiang Ye (Senior Member, IEEE) received the B.S. and M.S. degrees in mechanical and electrical engineering from the Harbin Institute of Technology, Harbin, China, in 1999 and 2001, respectively, and the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2006.

He has been a Professor with the University of Chinese Academy of Sciences, Beijing, since 2009, and was a Visiting Assistant Professor with the Institute of Advanced Computer Studies (UMIACS), University of Maryland, College Park, until 2013. He has authored or coauthored more than 100 articles in refereed conferences and journals including IEEE CVPR, ICCV, and PAMI. His research interests include image processing, visual object detection, and machine learning. He pioneered the Kernel SVM-based pyrolysis output prediction software which was put into practical application by SINOPEC in 2012. He developed two kinds of piecewise linear SVM methods that were successfully applied to visual object detection.

Dr. Ye received the Sony Outstanding Paper Award.