Enhancing Representation Learning with Spatial Transformation and Early Convolution for Reinforcement Learning-based Small Object Detection

Fen Fang, Wenyu Liang, Yi Cheng, Qianli Xu and Joo-Hwee Lim

***Abstract*—Although object detection has achieved significant progress in the past decade, detecting small objects is still far from satisfactory due to the high variability of object scales and complex backgrounds. The common way to enhance small object detection is to use high-resolution (HR) images. However, this method incurs huge computational resources which grow squarely with the resolution of images. To achieve both accuracy and efficiency, we propose a novel reinforcement learning framework that employs an efficient policy network consisting of a Spatial Transformation Network to enhance the state representation learning and a Transformer model with early convolution to improve feature extraction. Our method has two main steps: (1) coarse location query (CLQ), where an RL agent is trained to predict the locations of small objects on low-resolution (LR) (down-sampled version of HR) images; (2) context-sensitive object detection where HR image patches are used to detect objects on the selected coarse locations and LR image patches on background areas (containing no small objects). In this way, we can obtain high detection performance on small objects while avoiding unnecessary computation on background areas. The proposed method has been tested and benchmarked on various datasets. On the *Caltech Pedestrians Detection* and *Web Pedestrians* datasets, the proposed method improves the detection accuracy by 2%, while reducing the number of processed pixels. On the *Vision meets Drone object detection* dataset and the *Oil and Gas Storage Tank* dataset, the proposed method outperforms the state-of-the-art (SotA) methods. On *MS COCO mini-val* set, our method outperforms SotA methods on small object detection, while also achieving comparable performance on medium and large objects.**

***Index Terms*—Small object detection, Reinforcement Learning, Coarse-to-fine framework.**

I. Introduction

OBJECT detection is a fundamental functionality in numerous vision-based applications, such as surveillance, autonomous driving, remote sensing and mobile robots. Deep neural networks have facilitated constant updates of state-of-the-art (SotA) performance of object detectors on public datasets, such as PASCAL VOC [1] and MS COCO [2]. However, these datasets generally contain normal-size objects in the images. In contrast, detection performance is much lower on images with small objects. The deteriorated performance mainly comes from four factors. (i) The features that discriminate small objects from the background diminish due to multiple rounds of down-sampling operations by the convolutional layers in the backbone CNN. (ii) The bounding box of some small objects may be out of the scope of anchors or grids that are predefined according to the distribution of ground truth boxes in the training set. Information on such small objects is thus lost during the training stage. (iii) Since the scales of very small instances can vary in a wide range, the receptive field may not match the size of small objects, especially on LR images. (iv) There is a lack of significant benchmarks. Few existing benchmarks focus on scenarios where small objects are distributed in a large area, which is common for images shot from a long distance (e.g., aerial images). To address the first three factors, conventional methods usually resize the input images to different scales or reduce the down-sampling times in CNN to keep adequate discriminative features in the object detector training. However, increasing the resolution of input images (corresponding to the high dimension of feature maps) casts a heavy computational burden. On the other hand, the common way to address the last factor is to create a customized dataset of small objects, which requires intensive human labour, especially in the scenario of tiny object labelling in large images.

To reduce the computation cost and manpower needs, [5] proposes to query the coarse locations of small objects in input images during detector training. As depicted in Fig. 1 a, locations of small objects are queried in the training set so that only regions containing small objects will be trained using HR feature maps while the remaining background regions are in the original resolution. This alleviates the need to spend computation time on the background areas. After training, images in the testing set are directly processed by the detector. Another method, DZN [3] (Fig.1b), is proposed to first train coarse and fine detectors on LR and HR images, respectively, and then apply a reinforcement learning-based (RL-based) algorithm to query regions that have a higher likelihood of the presence of small objects. The query results from the last step are combined into the state for the next step query. The regions are zoomed in and scanned using the fine detector and the background areas are detected using the coarse detector.

All authors are with the Institute for Infocomm Research, A*STAR, Singapore.
This research is supported by the Agency for Science, Technology and Research, under the AME Programmatic Funding Scheme (Project#A18A2b0046).

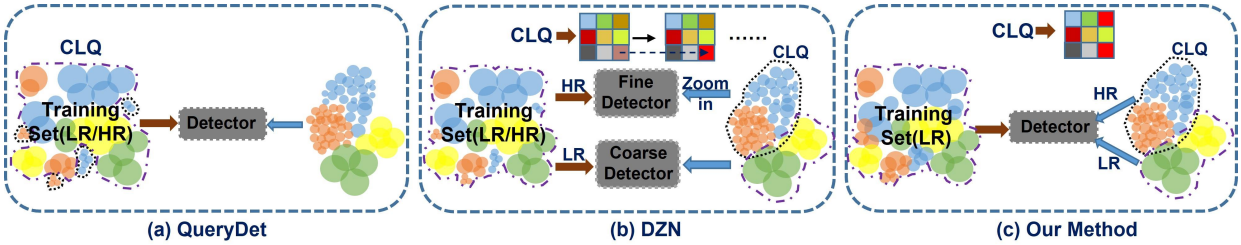

Fig. 1. Structure comparison among our method and two most related methods: (a) illustrates the rough pipeline of the QueryDet. It learns the CLQ in the detector training stage on LR or HR images. The testing images are directly processed by the detector. (b) depicts the learning procedure of the DZN. It trains two detectors, fine detector and coarse detectors, on high-resolution and low-resolution images, respectively. The CLQ is learned independently of the detector training stage in a sequential manner. Given an input image, the selected regions by the CLQ are zoomed in and fed to the fine detector, remaining areas are fed to the coarse detector. (c) shows that our method also learns the CLQ independently of the detector training stage. But our method only has one detector in LR images and trains the CLQ in a parallel mode with a single step. Hot regions in CLQ (e.g., red) indicate selected small object regions.

Notably, unlike [5] QueryDet where the query of locations of small objects happens in the detector training stage, this method implements a query of the coarse regions of small objects in an independent stage. In general, the performance improvement of the query learned in an independent stage is more significant compared to the query learned concurrently with detector training. This is because it is less affected by detector learning when domain shift occurs among different domains.

In this paper, we propose a novel framework that queries the coarse locations of small objects from predefined regions on HR images, independent of the detector's training stage, as illustrated in Fig. 1c. In the proposed framework, image regions are differentially processed before passing to a single pre-trained detector, i.e. regions predicted to contain small objects are fed into the detector in HR, while background regions (without small objects) are presented at LR. Moreover, the CLQ of small objects in the proposed method is learnt in a parallel mode, which means that all highly suspicious regions can be predicted by the CLQ simultaneously in a single step. This requires the strong discrimination ability of the CLQ algorithm. Hence, we train an RL agent using a novel policy network to implement CLQ. The policy network consists of an STN to achieve better state representation and combines Transformer and CNN by taking the advantage of CNN: local low-level feature extraction for early visual processing, and the advantage of Transformer: global feature extraction and self-attention mechanism for localizing regions of small objects [12]. A simple example in Fig.2 demonstrates the pipeline of the proposed method. Given an LR input image, the CLQ algorithm predicts the locations of small objects. Remarkably, the locations of small objects (e.g., person) and the background areas (with grey mask) containing normal-size objects (e.g. vehicles) are detected by the detector in high resolution and low resolution, respectively.

The main contributions of this work are:

- We introduce a novel pipeline that employs CLQ for small object regions in a parallel manner, independent of object detector training;
- We propose to use an STN to enhance state representation in the policy network;
- We design a novel policy network with early convolution and Transformer for accurate and efficient CLQ implementation.

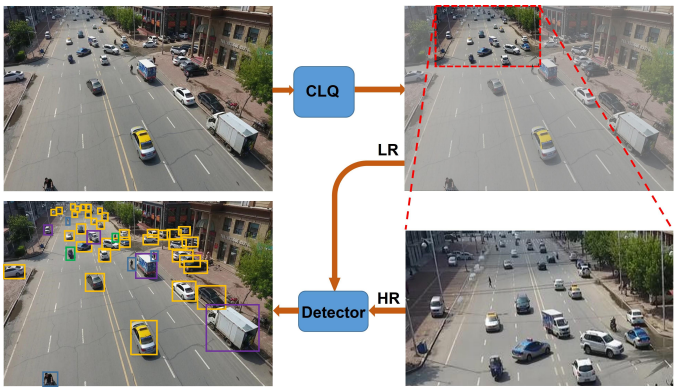

Fig. 2. Illustration of our approach using an example from the VisDrone-Det dataset. The input is an LR image, after querying coarse locations of small objects by the CLQ algorithm, the high-resolution confirmed coarse locations and the low-resolution background area (with grey mask) are proceeded to the object detector to output the final detected boxes.

In this paper, we will use several acronyms to refer to key terms and concepts. For clarity and ease of understanding, we define each of these acronyms below:

- CLQ: Coarse Location Query.
- MLP: Multi-layer Perception.
- STN: Spatial Transformation Network.
- EVP: Early Visual Perception.
- PE: Patches Embedding.
- PSG: Parameterised Sampling Grid.
- OFM: Output Feature Map.
- MDP: Markov Decision Process.
- DQN: Deep Q-learning Network.
- IoU: Intersection over Union.
- FLOPs: Floating Point Operations.
- AP: Average Precision.
- P_{pn}: Processed Pixel Number Percentage.
- TP: True Positives denoting the set of correctly detected objects.
- FP: False Positives meaning the set of falsely detected objects.
- FN: False Negatives referring to objects missed by the

detector.
- CPD-WP: Caltech Pedestrian Detection Dataset and Web Pedestrian Dataset.
- VisDrone-Det: Vision meets Drone Object Detection Dataset.
- OGST: Oil and Gas Storage Tank Dataset.

II. Related Works

Existing small object detection methods fall into two groups: scale-ware method and HR input-based method.

A. Scale-Aware Method

Using different scales of images to train multiple detectors is an intuitive approach to improve small object detection accuracy, as each detector can achieve the best accuracy on objects in a certain range of scales. But this approach incurs high computation costs. To deal with the computation cost issue, some works [6], [7], [8] deploy feature-level pyramids of different scales in a single detector to reduce the computation cost. For example, The Sniper [8] handles ground truth objects with context regions at the appropriate scales. The single shot detectors (SSDs) [9], [10], [11] apply multi-scale features to alleviate accuracy drop caused by skipping region proposal network for cost reduction. Especially, to enhance the detection of tiny objects from UAV images, FS-SSD [11] fuses the scaling-based SSD and feature formed by adding an extra scaling branch of the deconvolution module with an average pooling operation. The feature pyramid networks [13], [14], [15] design a top-down structure to fuse feature information of different scales to improve object detection accuracy. The RFB enhances the feature discrimination and robustness of the object detection with various sizes by using the relation between scale and eccentricity of receptive fields. The TridentNet [16] generates scale-specific feature maps by applying a multi-branch framework with different receptive fields in a parallel way. The S3fd [17] uses a scale-equitable method to enhance the effectiveness of small face detection. The SBL [18] utilize a re-weighting strategy to significantly enhance the detection accuracy on small objects. Instead of simply fusing object predictions from multi-scale feature maps, the STDN [19] is equipped with an embedded scale-transfer layer to explore the inter-scale consistency across multi-scale. The scale-match [20] approach aligns the object scales between datasets for the pre-training and the one for detector training to improve performance on tiny person detection. The latter is more efficient as its "focus" operation is conducted on feature pyramids other than image pyramids, which can reduce redundant computation.

B. High-Resolution Input-Based Method

Using HR images as input is another common method to fulfil the requirement of accurate detection of small objects. For example, the SRCNN [21] and the VDSR [22] enhance detection accuracy on super-resolution images by using very deep neural networks. ChaDeConv [23] develops a channel-aware deconvolution layer across very deep layers to strengthen the small object detection recall rate. The main problem associated with the methods is the heavy computation cost of HR imagery for large regions. To alleviate this problem, some work [3], [4], [5], [24], [25], [26] employ RL to find regions containing intensive small objects. For instance, DZN [3] utilizes the RL algorithm to sequentially select regions of small objects to zoom in. These regions and remaining regions are processed using a fine detector (trained on HR images) and a coarse detector (trained on LR images), respectively. In [55], a more straightforward approach is proposed to use tiling to partition high-resolution images into fixed-size grids, which are then cropped out and fed as input to the detectors. This method is brutally simple, as it considers all grids as potential small object regions. In [25], an RL agent trained with a dual reward setting is also used to adaptively choose the spatial resolution in each image. Moreover, AutoFocus [4], ClusterDet [53], DMNet [54], and QueryDet [5] employ a mechanism that first predicts the rough locations of small objects on LR images and then perform object detection from the queried rough locations in HR images. Notably, ClusterDet, DMNet, and QueryDet share a similar framework. In addition, various pre-processing techniques can be employed to improve small object detection, such as Spectral clustering [48], [49] and concept relevance mining [50].

The works that are most related to ours are the DZN [3], ClusterDet[53], DMNet[54] and the QueryDet [5]. Since the last three works share a similar approach and QueryDet is the most recent among them, we have chosen QueryDet as a representative for the comparison. Our method shares a similarity with DZN in that both employ CLQ algorithms based on reinforcement learning, which are implemented independently of the detector training stage. But the differences (shown in Fig.1) are that (i) the DZN uses coarse (trained on LR images) and fine detectors (trained on HR images) for the detection of background areas and regions of small objects, while our method only uses a single detector; (ii) the CLQ algorithm, namely RL agent, is learnt in a sequential manner (multiple queries) in the DZN while the one in our method is in a parallel way (single query). According to [26], in multiple queries, there exists uncertainty about when to stop the query in the inference stage without providing ground truth small objects region information. Thus, the single query is more efficient and reasonable. Moreover, an enhanced RL policy network is designed combining an STN, Transformer and CNN to boost the CLQ accuracy in our method. As mentioned in Section I, the main difference between our method and the QueryDet is that the QueryDet learns the CLQ of small objects in the detector training stage, namely, the CLQ is integrated into the detector. In this case, if the distributions of object scale vary a lot between the training set and the testing set (e.g., the two sets are from different datasets), the QueryDet is not able to help with performance improvement. In contrast, our method treats the trained detector as a 'black box', and adjusts the small object scale in the input image to cater for the preference of the detector by taking out the regions selected by the CLQ

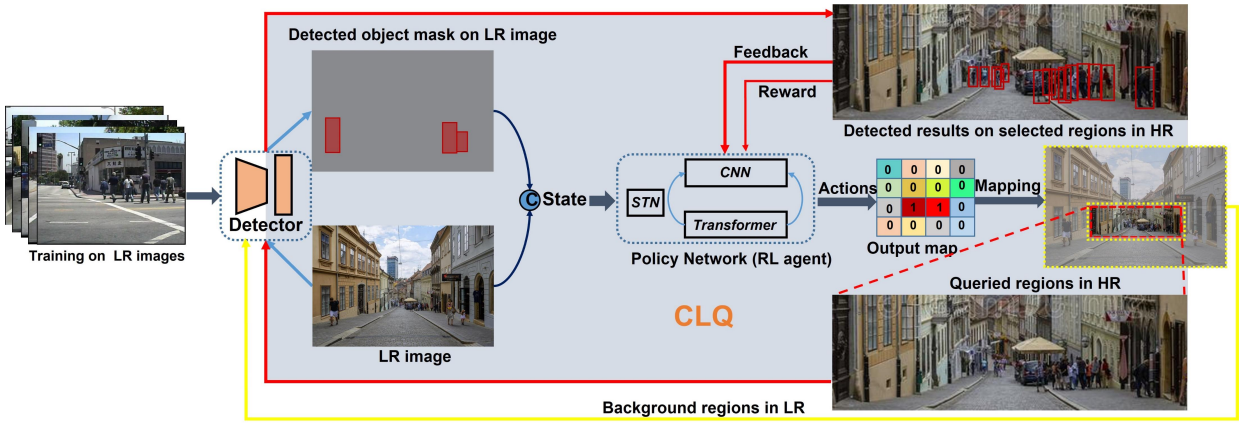

Fig. 3. The overall framework of our method. First, The detector is trained on LR images. Given an LR image, the detector scans it to generate detected bounding boxes of objects, and then all bounding boxes are highlighted on the mask of the image. Then, the testing image and the mask are concatenated as a state to the RL policy network which is in an STN, transformer and CNN collaborative structure. The policy network gives actions on all predefined regions to create an output map with values of '0' and '1' corresponding to background areas and small object areas. The selected small object areas (within dashed red boxes) are scanned by the detector in HR, and the background areas (within dashed yellow boxes) are detected in LR. The detection results generate the reward value and give feedback for updating policy network parameters. The RL policy network (RL agent) gives action prediction on each image in a single step.

in HR. Hence, even though the testing and training sets are from different datasets, our method can still boost the detection performance.

III. Problem Formulation and Background

The CLQ is the key part of the proposed framework. In this section, we show the formulation of the CLQ problem and the intuition of our framework design.

A. Problem Formulation

The works [51], [52] formulate both searching-agent learning and cutting-agent learning as MDP which is a popular system for modelling an agent to make a sequence of decisions, with the goal of localizing optimal regions for object detection from images and object segmentation from videos respectively. As the CLQ aims at selecting locations of small objects from predefined regions of an image simultaneously, which is similar to the goals of the searching-agent [51] and cutting agent [52], we also formulate the CLQ task (as shown in the light blue shaded area in Fig.3) as a single-step episodic MDP. Then, the RL algorithm is used to solve the problem. A standard RL is composed of a set of actions A, a set of states S, a policy $\pi(a_t|s_t)$ with $a_t \in A$ and $s_t \in S$ that is produced by the policy network on action selection given a state, a state transition function P, and a reward function R.

Given each input image, the system observes the current state, makes an action decision and computes the reward from this decision, then conveys the feedback to optimize the policy network (toward optimal policy π^*) for maximizing long-term reward. DQN is employed in this work as the RL algorithm to train the policy network. Hence, the policy network optimization process on state s_t can be represented by

$$Q_\pi(s_t, a_t) = \mathbb{E}[R_t|s_t, a_t]. \tag{1}$$

According to the Bellman optimality equation [27], given a state, the current Q-value after taking an action can be computed by (1), and the optimal reward of taking the action can be approximated using the sum of the immediate reward and the discounted reward at the next state depicted in (2).

$$Q^*(s, a) = \mathbb{E}[r + \gamma \max_{a'} Q^*(s', a')] \tag{2}$$

where $s' = P(s)$ is the next state and $a' = \pi(s')$. $\gamma \in [0, 1]$ is the discount factor used for balancing the importance of the current and future rewards.

The CLQ problem can be then converted to an optimization problem of the policy network parameter θ toward θ^*, so that the current Q-value is approximately equal to the optimal Q-value (e.g. $Q_\pi(s, a; \theta^*) \approx Q^*(s, a)$). Under such circumstances, after the DQN model training, the optimized policy network is able to select an optimal action on each state to return maximized user-defined reward.

As motioned in Section I, to efficiently learn the optimal policy parameter θ^*, we design a policy network combining Transformer and CNN and propose a single-step multi-action RL agent learning protocol.

B. Background and Intuition

In this study, the RL-based CLQ algorithm is learned on LR images. This incurs a problem that too much resolution compression results in the loss of important information of small object clusters localization (e.g. LR images may be blurred). To accurately query small object regions, the policy network should be strong on discriminative feature extraction. Motivated by work [28] that great achievement of a hybrid Transformer CNN network on image denoising and work [29] that excellent performance of bidirectional transformer network for video deblurring, we design a combined Transformer-CNN policy network to boost CLQ's accuracy.

Referring to the work presented in [30], the objects can be split into three size ranges according to the height of the

bounding box of each instance: small-with range of fewer than 50 pixels, medium-with range from 50 to 300 and large-with range above 300. Although the statistical results reported in [30] show that the percentage of a small object in MS COCO [2] is 0.43, here are only around 10 out of 80 classes with more than 50% of small instances. This means, in the common objects dataset, most of the classes have enough samples in normal size. Moreover, to ensure detection accuracy on objects of medium and large sizes and save computation cost and depth of neural networks, the object detector is trained on the down-sampled version of HR images. In our method, as long as the CLQ performs well in locating small objects, the final object detection accuracy can be improved because there is no performance loss on large objects.

IV. Our Method

A. Overall Framework

The overall framework of our method is illustrated in Fig. 3. It mainly consists of a detector and an RL agent. Basically, the detector training on the LR images is straightforward, which training detail will be elaborated in the experiment section. In this section, the focuses are on two parts that contribute to the performance improvement of the CLQ (i.e., highlighted by the light blue background in Fig. 3): (i) a novel design of policy network combining STN, Transformer and CNN, and (ii) a single-step multi-action RL agent training protocol. As can be seen from Fig. 3, once an LR image is fed to the detector, a mask with detected boxes is generated. Then, it is concatenated with the input image along the channel dimension as the state of the RL agent training. The RL agent predicts actions from corresponding predefined regions, and the output map is then yielded. Then, the mapping from the output map to the LR image is performed to extract small object regions and which are fed to the detector in HR to scan the small objects. The RL agent receives scanned results as feedback to update the policy network's parameter to obtain a higher reward value from the next action. In each epoch (the training goes through all images), as mentioned earlier, the RL agent learning on each image is in a single step. By training the RL agent systematically in this fashion, the accuracy of the CLQ algorithm can be boosted, consequently, the overall detection performance is enhanced.

B. Policy Network

When we design a system for an object recognition task, the robustness of the system to input variations is the most desirable property [31]. The input variations include object scale variation, viewpoint variation, object deformation and so on. They affect the performance of object recognition tasks especially when the objects are very small in images. The policy network in our framework is designed to accurately predict regions with small objects regardless of object scale and viewpoint. The detailed architecture of the policy network is depicted in Fig.4. There are three main components: (i) a STN[1]

[1]The original paper [31] uses the term spatial transformer. To avoid confusing it with modern Transformer [32], this research uses spatial transformation instead when referring to the method [31]

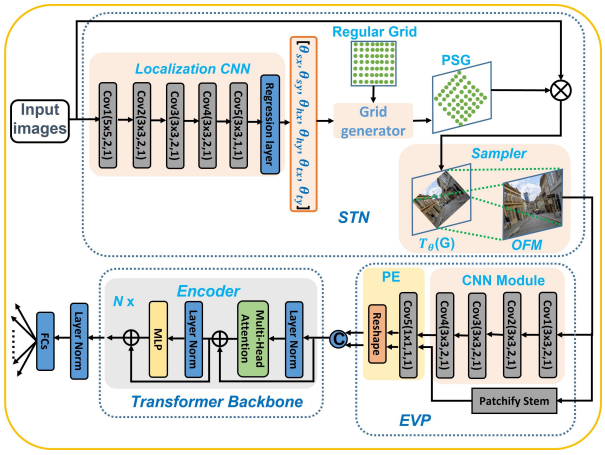

Fig. 4. Detailed architecture of the policy network, which contains three main components. The first is STN with a localization CNN to generate transformer parameters θ, a grid generator for PSG generating from the regular grid of output feature map and a sampler to produce OFM from sampling points ($T_\theta(G)$) using bilinear interpolation. The second is the EVP block for producing visual patches from the transformed output of STN. The last one is the transformer block to predict final actions.

that consists of a localization CNN to predict transformation parameters θ, a grid generator to yield PSG and a sampler to generate an OFM from sampling points $T_\theta(G)$ along with the input image. (ii) an EVP that is composed of a CNN module and patchify stem for 'patches' producing followed by a patch embedding layer, and (iii) a transformer block to predict actions on each pre-defined area in input images.

1) STN: The spatial transformation model is a visual attention mechanism that was proposed in [31] to counter the spatial transformation variations in input images by embedding explicit spatial capabilities in CNN. In our method, we adopt the STN mechanism to generate a feature map with enhanced representation learning to benefit the EVP and Transformer components. Compared with the original input image, the feature map contains richer and more sensitive information. This is because the spatial arrangement of pixels retains crucial information about the input content, especially when object scales and view angles vary in a wide range.

In our method, the localization CNN contains five convolutional layers and a regression layer. These convolutional layers act as feature extractors, each layer is with a kernel size of 3×3 and padding of 1, and the stride is 2 for the first four layers and 1 for the last layer. The regression layer is to predict the transformation parameters σ.

We consider three transformations-scaling, translating and shearing on the x and y axes, thus the localization CNN outputs six parameters (as shown in Fig.4) where pairs (σ_{sx}, σ_{sy}), (σ_{hx}, σ_{hy}) and (σ_{tx}, σ_{ty}) represent scaling, shearing and translation along x and y axes, respectively. The transformation matrix T is defined as

$$T = \begin{bmatrix} \sigma_{sx} & \sigma_{hx} & \sigma_{tx} \\ \sigma_{hy} & \sigma_{sy} & \sigma_{ty} \end{bmatrix}, \tag{3}$$

Then, the pointwise transformation, namely, the grid generator, from the input image (source) to the output feature map

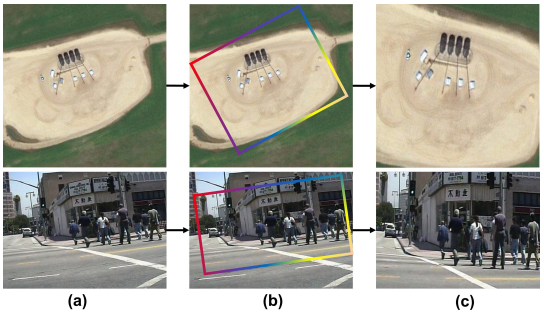

Fig. 5. Two example images where an STN is applied to produce feature maps: (a) The inputs to the STN. (b) Applied the predicted transformation to the input images. (c) the output feature maps (OFM) of the STN.

(target) can be computed as

$$\begin{bmatrix} x_{ij}^s \\ y_{ij}^s \end{bmatrix} = \begin{bmatrix} \sigma_{sx} & \sigma_{hx} \\ \sigma_{hy} & \sigma_{sy} \end{bmatrix} \cdot \begin{bmatrix} x_{ij}^t \\ y_{ij}^t \end{bmatrix} + \begin{bmatrix} \sigma_{tx} \\ \sigma_{ty}^s \end{bmatrix}, \tag{4}$$

where the pair (x_{ij}^t, y_{ij}^t) is the target coordinates of the regular grid in the output feature map. The pair (x_{ij}^s, y_{ij}^s) is the source coordinates in the input image, it indicates indices where we should sample in the input image to get the desired transformed output feature map. These indices, as shown in Fig.4, form the PSG.

The last step is to generate the output feature map. The sampler traverses over the entries of the PSG to look up the corresponding pixel values from the input image using bilinear sampling. In the sampling process, the pixel value of the sample point (i, j) is computed using the closest four neighbouring points-upper_left $(i-1, j-1)$, upper_right $(i+1, j-1)$, lower_left $(i-1, j+1)$ and lower_right $(i+1, j+1)$. Then we can derive the pixel value at point (i, j) $v_{i,j}$ in the output feature map as

$$\begin{aligned} V_{i,j} = & U_{i-1,j-1} \cdot d_i \cdot d_j \\ & + U_{i+1,j-1} \cdot (1-d_i) \cdot d_j \\ & + U_{i-1,j+1} \cdot d_i \cdot (1-d_j) \\ & + U_{i+1,j+1} \cdot (1-d_i) \cdot (1-d_j), \end{aligned} \tag{5}$$

where $U_{i,j}$ is the pixel value at point (i, j) in the input image, d_i and d_j denote the horizontal and vertical distance from the sample point to the border of the cell formed by the four closest neighbouring points. After traversing all sample points, we obtain the output feature map.

We show two visualized results of multiplying out the STN predicted transformations in Fig.5. The input image in the first row is from OGST, the objects to be recognized are oil and gas tanks. The input image in the second row is from CPD, and the objects to be processed are pedestrians. The output (OFM) of the STN illustrates how the network is adept at 'cropping' and 'rescaling' specific parts of the input images that are relevant to the task at hand. By focusing on these critical areas, subsequent components of the network can prioritize the most informative features in their processing. This echoes the reason for adopting STN in our framework, where it simplifies the following *EVP* and *Transformer* components.

2) EVP: Benefiting from the self-attention mechanism, Transformer equipped with global modelling ability has achieved great success on vision tasks [32], [33], [34] in recent years. In a standard visual transformer (e.g., ViT [32]), the input of the encoder is linearly embedded in non-overlapping patches from the patchify stem implemented by a large kernel and large stride convolution. It has been demonstrated that large kernel and large stride convolution in visual Transformer cause convergence challenges (e.g., the training converges slowly) and optimization challenges (e.g., optimization is very sensitive to the learning rate and weight decay choice). Because CNN is easy and robust to train and good at low-level local feature extraction, a CNN module proposed in [35] is employed to replace the patchify stem so as to address the aforementioned challenges. The CNN module usually consists of multiple layers of small stride and small kernel convolutions, which inject inductive bias into the encoder to enhance the performance of the Transformer model. It is reported that the large kernel convolution is very useful on small feature maps, and is particularly powerful on downstream tasks (e.g., classification, detection) [36]. As mentioned earlier, the patchify stem works similarly to a large kernel convolution layer (e.g., generating a 32×32 patch size). Therefore, to benefit from both CNN for low-level feature extraction and large kernel convolution, in the EVP component, we consider using both the conventional CNN module and the patchify stem to generate 'patches'. The two patch vectors will be concatenated as the input to the encoder after PE.

As can be seen in Fig.4, the CNN module consists of four convolutional layers, each layer is with a kernel size of 3, a stride of 2 and a padding of 1. The Patchify stem is a 16×16 convolution operation. The PE block consists of a convolutional layer with 1×1 kernel and 1 step stride, followed by a reshape operation.

3) Transformer Backbone: The Transformer backbone contains N stacked encoders, where $N = 4$ in this work. In each encoder, there are two parts: a multi-headed attention mechanism and a 2-layer MLP. A layer normalization and residual connections are applied before and after the two components, respectively. The layer normalization is to stabilize the hidden layer's dynamics and to save training time. The residual connections provide an alternative path for data to reach the latter parts of the network by skipping some layers, it helps to solve the vanishing gradients problem and to converge much more easily during the training process.

At the end of the policy network, there are two fully connected layers to linearly convert the feature vector to the output vector.

C. Single-Step Multi-Action RL Agent

Another contributor to boosting CLQ's accuracy is the single-step multi-action RL agent training protocol. Given an input image, there may be multiple regions containing small objects. Thus, as illustrated in [3], sequential actions are taken to select all regions. Implementation of all regions selection in a single query relies on our action design. In this section, we describe the main components in RL agent training: *action*,

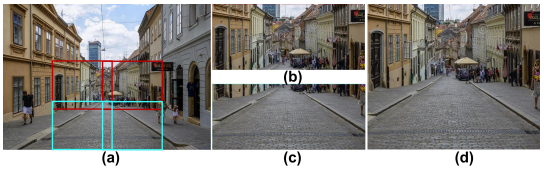

Fig. 6. An example of different regions query on an input image. (a) An LR input image. (b) selected regions in red boxes. (c) Selected regions in blue boxes. (d) Regions in all boxes. By applying the proposed reward function, the RL agent is encouraged to select the regions in red boxes.

state representation, *environment*, *reward* and *policy network optimization*.

Action. Our RL agent simultaneously analyzes all regions from an image in a single step. In this scenario, an action refers to deciding whether a region contains a small object or not. Hence, the number of *action*, n_a, is equal to the number of predefined regions and the space of each action is 2 (e.g., '0' for regions without any small object, '1' for regions containing small object). Thus, the total action space is $2 \times n_a$. While the number of predefined regions depends on the dimension of the LR images. Assuming that the LR image in the dimension of $W \times H$, and all predefined regions are in the same dimension (w_r, h_r), the overlap between two adjacent regions is δ, the layout of regions in the LR images is $n_w \times n_h$, then, the dimension of each region can be determined as $w_r = \frac{W+(n_w-1)*\delta}{n_w}$, $h_r = \frac{H+(n_h-1)*\delta}{n_h}$. The number of actions is computed by n_a=n_w*n_h, which is a large value, and thus action space is even larger. However, training the policy network with a large action space has issues where the process can be unstable and takes a long time to converge. To overcome these issues, a novel policy network is designed.

After executing an action a, the potential small object areas can be represented by

$$A_{small} = \sum_{i=1}^{n_a} (c_{x_i}, c_{y_i}, w_i * a_i, h_i * a_i), \tag{6}$$

where (c_{x_i}, c_{y_i}) define the center point of the i^{th} region, and w_i and h_i are width and height of the region, a_i is the action decision on the region. The system then will acquire A_{small} in HR and feed it to the detector for small object detection.

State Representation. As shown in Fig.3, the state representation encodes two channels of information: 1) the input image in LR; and 2) A mask map sharing the same width and height as the input image. The mask is generated by first drawing the bounding box of the detected objects, and then filling the boxes with a single color (e.g., red) weighted by the confidence score while filling the other areas with grey. The two channels of information are concatenated to form the state representation. Such a state can help the policy network avoid querying regions containing many filled boxes so that the training converges faster and the RL agent performs more accurately.

Environment. An environment is where the RL agent collects feedback after taking action and interacting with it. In our framework, the environment is the detection of queried HR regions. The detected results are conveyed to the policy network to drive it to seek ways to maximize the reward.

Reward Function Modeling. Our proposed framework aims at querying small object regions in an LR image and acquiring HR of the regions for small object scanning using the detector. Thus, the desired reward function should reflect both the final detection accuracy and running time efficiency. In other words, the reward function encourages the RL agent to query regions that contain more small objects, while keeping the number of selected regions low. The reward function on selected n_q regions by the CLQ is hence defined as follows,

$$R(s, a) = R_{det}(Y_{LR}, Y_{HR}) - R_{rtc}(a), \tag{7}$$

$$R_{det} = IoU(Y_{LR}, Y_{gt}) + \sum_{i=1}^{n_q} IoU(Y^i_{HR}, Y_{gt}), \tag{8}$$

$$R_{rtc} = \lambda_{LR} + (\lambda_{HR} - \lambda_{LR}) \frac{\sum_{i=1}^{n_a} a_i}{n_a}, \tag{9}$$

where R_{det} denotes the reward on detection accuracy and R_{rtc} represents detector running time cost. IoU is the intersection over the union of two regions, which is a commonly-used metric in object detection. Y_{LR}, Y_{HR}, and Y_{gt} are detected results on LR regions, HR regions, and ground truth regions, respectively. λ_{LR} and λ_{HR} are coefficients of running cost of detector on LR and HR regions, their specific values are determined by the proportion of the running cost in the total reward, as shown in 7.

An example of how the reward function affects the RL agent on action selection is shown in Fig. 6. Given the input image in Fig.6(a), there are three combinations of small object region selection, which are regions in two red boxes, in two blue boxes, and in all four boxes. It can be found that (i) regions (see Fig. 6(d)) in all boxes contain all small objects (e.g., tiny person), (ii) regions (see Fig. 6(b)) in red boxes have the vast majority of small objects inside, (iii) regions (see Fig. 6(c)) in blue boxes only include a little part of small objects. If we only consider the final detection accuracy (R_{det}), all four regions should be selected to acquire HR. However, doing so means that about half of the time is wasted in processing HR images on 'barren' regions (blue boxes) which lead to a minimal gain in accuracy. By considering both detection accuracy and running time cost, our reward function design encourages the RL agent to select regions in red boxes to acquire HR.

Policy Network Optimization. As mentioned in Section III-A, we formulate our problem, the CLQ learning, in a DQN framework, which estimates the long-term reward for actions by optimizing a policy network. Given a state, the current Q-value after taking an action can be computed in (1), and the optimal Q-value can be approximated by (2). To learn an optimal policy yielding the maximized reward on action on the i^{th} region, a_i, the parameter θ of the policy network on this action can be learnt by minimizing the loss between the optimal Q-value and current Q-value in (10). As there are n_a actions taking on a state, the total loss across all actions on a state is computed by (11).

$$L_{a_i}(\theta_t) = [R(s, a_i) + \gamma max_{a_i'} Q(s', a_i'; \theta_{t+1}) - Q(s, a_i; \theta_t)]^2. \quad (10)$$

$$L_{total}(\theta_t) = \frac{1}{n_a} \sum_{i=1}^{n_a} L_{a_i}(\theta_t). \quad (11)$$

After training the policy network, θ^* is obtained, the $Q^*(s, a)$ can be approximated by $Q(s, a, \theta^*)$, and thus an action can be obtained by

$$a_t = \arg\max_{a \in A_t} Q(s_t, a_{t-1}; \theta^*). \quad (12)$$

When the action on a region is predicted as '1', then the system queries the region in HR for object detection. Otherwise, the regions with action predicted as '0' are regarded as background, and then they will be scanned by the detector in LR. Detection results on all regions will be merged to get the final prediction on the input image.

V. Experimental Setup

A. Datasets

Three samll object datasets (of different applications) are used to conduct the experiments: (i) CPD-WP [37] for pedestrian detection, (ii) VisDrone-Det [39] for multiple objects detection in UAV images and (iii) OGST [40] for oil and gas storage tank detection in remote sensing images. To demonstrate the effectiveness of our method in detecting small objects while also retaining the ability to detect larger ones, we conducted experiments on the MS COCO mini-val dataset [2]. This dataset comprises a diverse range of images that feature small, medium, and large objects, as well as human subjects, in various everyday contexts.

1) CPD-WP: The CPD dataset contains densely annotated frames from street-view videos. The frames are annotated with pedestrians in the dimension of 640×480. There are 4321 images in the training set and 4088 images in the testing set. As 640×480 is already a low dimension, down-scaling the dimension may result in poor-quality images. Thus, on one hand, we set LR images as the same as the HR images in the original dimension. In this way, the regions selected by the RL agent are only resized without using HR, namely, zoom-in is implemented on the potential regions of small objects. On the other hand, to validate the effectiveness of our method on datasets with varying resolutions, similar to [3], we construct WP dataset by searching 100 HR images from YFCC100M [38] dataset. Each of the original HR images contains varied sizes and densely distributed pedestrians. The HR images are re-scaled to 1600 pixels along the longer side for saving GPU memory. Then, down-sample the HR images to the dimension of 640×480 to create LR images of WP dataset. Hence, the detector trained on CPD training set is evaluated on CPD testing set and WP dataset. In the RL agent training for the CLQ, the predefined regions are in 236×180, and the action number on each image is then determined as 9.

2) VisDrone-Det: The dataset is composed of images taken by drone for multiple classes (e.g., the pedestrian, person (not walking), car, van, bus, truck, motorcycle, bicycle, awning-tricycle, and tricycle) object detection. It has 6471 images for training, 548 images for validation and 3190 images for testing. Among all testing images, 1580 images are for competition and 1610 images are for research benchmarks which are used in benchmarks of this work. The image dimension in the dataset is not fixed (e.g., 1920×1080, 960×540), we reduce the dimension of all images in the validation and testing set to 768×432 to collect LR images. The original images are regarded as HR images. The dimension of the predefined regions on LR images is 216×160, and the number of actions is 12.

3) OGST: The dataset contains 30 cm resolution remote sensing RGB images, which are cropped into 512×512 tiles to establish HR images, each tile has at least one tank. Following [40], the LR images are obtained by down-scaling the HR images $\times$ 4, and therefore, the dimension of LR images is 128×128. The dataset is split into training and testing sets containing 760 and 152 images, respectively. The predefined regions have a dimension of 50×50. The number of actions is thus 9.

4) MS COCO mini-val: Our method was evaluated and benchmarked against other approaches on MS COCO mini-val set consisting of 8,059 images in dimension of 600×400. Due to the similar image resolution between this dataset and CPD, which is not very high, we have chosen to first resize all images to dimension of 640×480, then adopt the same setting as CPD for HR and LR images in this dataset.

B. Detector and RL Agent Training

We ran three detectors to verify the object detection performance of our method. The first two are the one-stage detectors, SSD [9] and Yolov7 [41], and the third one is the two-stage detector, Faster R-CNN (FRCNN) [42]. All detectors and the RL agents are trained on LR images for all datasets. For some commonly used large dataset, such MS COCO, to save time on detector training and ensure optimal detector performance, we downloaded pre-trained Faster-RCNN, Yolo-v7, and SSD detectors trained on the training set.

The RL agent is trained using a DQN framework. We initialize all weights of the DQN model randomly with respect to a normal distribution and employ Adam optimizer [43] with a learning rate of 10^{-4} to optimize the model. During training, we adopt a prioritized experience replay policy [44], [45] to sample important transitions more frequently. We train the DQN model with a batch size of 64, set the size of the replay buffer as 10,000, and initialize the optimal Q-value and the current Q-value randomly. To balance exploration and exploitation, we adopt ϵ-greedy policy [46]. After every 20 episodes, we use the current Q-value to update the optimal Q-value. The policy network with parameters, θ, on all actions is jointly optimized by minimizing the loss in (11). The coefficients λ_{HR} and λ_{LR} in the reward function expressed in (9) are set as 1 and 0.1, respectively. The discount factor, γ, representing the effect of the distant future reward on

immediate reward is 0.99. The training epochs on all datasets is 1000.

C. Metrics and Benchmark Baseline

The AP and the processed pixel number percentage P_{pn} are used as the evaluation metrics.

Here, AP is to evaluate the final object detection results, and it is obtained by computing the area under the precision-recall curve with the Precision and Recall defined by

$$Precision = \frac{TP}{TP+FP}, \ Recall = \frac{TP}{TP+FN}, \tag{13}$$

An object is regarded as correctly detected if the *IoU* between the detected box and the ground truth box is over a threshold (e.g., 0.5 used in this work). Ten *Precision* values can be obtained by varying the threshold from 0.5 to 0.95 with a step size of 0.05, which can be averaged as *AP*. In the case of multi-class object detection, *AP* is computed class by class; and the mean *AP* of all classes is computed as the final metrics. In addition, we have also included metrics *AP_S* (AP on small objects), *AP_M* (AP on medium objects), and *AP_L* (AP on large objects) for the VisDrone-Det and MS COCO mini-val datasets. On the CPD-WP dataset, we have focused on demonstrating the performance of our method on domain shifts from CPD to WP, thus, we have chosen only show *AP*. Since the OGST dataset mostly contains small objects with very few middle and without large objects, we have chosen to only report the *AP* for this dataset.

P_{pn} is to indicate the computational cost, it is computed as

$$P_{pn} = \frac{pix_{processed}}{pix_{HR}}, \tag{14}$$

where $pix_{processed}$ is the number of pixels that have been processed and pix_{HR} is the number of pixels of an image in HR.

We benchmark our method with four baselines on the two metrics. When the detector is trained on the LR images, it is called D_L. The first baseline ($Baseline\_1$) is to directly perform the evaluation of D_L on testing images in HR. The second baseline ($Baseline\_2$) is to scan all predefined regions of testing images using D_L in HR sequentially. Alternatively, the detector is trained on the HR images, which is called D_H. The third baseline ($Baseline\_3$) is to perform the evaluation of D_H on the testing images in LR. Finally, the last baseline ($Baseline\_4$) is to evaluate D_H on the testing images in HR.

Besides that, the performance of our method is also compared against the two most related methods DZN [3] and QueryDet [5]. Refer to Fig. 1 and section II-B for the difference between our work and the two works (e.g., the CLQ is learned in the training stage for QueryDet, but in the inference stage for DZN and our method; the CLQ is learned in a sequential mode for DZN but in a parallel mode for our method). Notably, input to our method, the DZN and the QueryDet are the LR images in the inference stage, and only the CLQ-selected regions are replaced in HR for small object detection.

To provide an intuitive demonstration of the effectiveness of our method, we also report the average inference time for each method using the three detectors in our evaluation. It should be noted that the inference time may vary depending on the development platform.

D. Platform Setup

All detectors are trained on all datasets using a machine with NVIDIA GeForce RTX 2080 GPU, and the RL agents are trained and tested on a machine with NVIDIA GeForce GTX 1080 GPU. All algorithms are run using the PyTorch framework.

VI. Experiment Results

On the testing set of each dataset, 5-fold cross-validation is used. In each fold, half of the images are randomly selected to train the RL agent (CLQ), and the remaining images are used for evaluation. The final evaluation results are the mean value of results from all folds.

A. CPD-WP

Table I shows the performance comparison among our method, the baselines and the most related methods on CPD-WP. As can be seen, except FRCNN detector on CPD, our method delivers the best results in terms of AP. On CPD, because the HR and LR images are the same, the results of $Baseline\_1$, $Baseline\_2$, and $Baseline\_4$ are the same. The AP of $Baseline\_2$ is the lowest on CPD. This is because while scanning all regions makes it possible to detect more small objects, it incurs many false detection after zooming in all regions. On WP, among the four baselines, $Baseline\_1$ and $Baseline\_4$ use the same detector and inputs, and thus their results are the same. $Baseline\_2$ gets the best results because the inputs are the regions where the distribution of object scale is closer to that in the training set (CPD). $Baseline\_3$ returns the worst results which suggests that naively scaling the input images can significantly hurt the performance. Among DZN, QueryDet and our method, on CPD, our method performs slightly better because of the carefully designed policy network. On WP, our method defeats DZN and QueryDet by around 2% and over 11%, respectively. QueryDet shows significantly inferior performance because its CLQ is learnt on the training set (CPD), and the object scale in WP is different from CPD, making the CLQ ineffective in predicting small object regions properly.

Regarding P_{pn}, because LR and HR images of CPD are the same, P_{pn} of all methods are the same, i.e. 100% except that of the $Baseline\_2$, which is 124% because of overlap of two adjacent regions. On WP, $Baseline\_1$ and $Baseline\_4$ take HR images as input. Therefore, their P_{pn} are the same. The P_{pn} of $Baseline\_2$ is slightly higher as there is overlap among predefined regions. $Baseline\_3$ has the lowest P_{pn} as it uses LR images as input. For DZN, QueryDet and our method, LR images are taken as the input and the regions selected by the CLQ are replaced in HR. Thus, the pixel number percentages are between the detection results purely on LR images and on HR images.

TABLE I

DETECTION PERFORMANCE COMPARISONS IN TERMS OF $AP \uparrow$, $P_{pn} \downarrow$ AND INFERENCE TIME↓ (IN SECOND) ON THE CPD AND WP DATASETS USING SSD, YOLOV7 AND FRCNN DETECTORS. AS THE LR AND SR IMAGES OF THE CPD DATASET ARE THE SAME, THE DETECTORS D_L AND D_H ARE THE SAME. THUS, DUE TO THE SAME DETECTORS AND THE SAME INPUTS, ON CPD, THE RESULTS OF $Baseline\_1$, $Baseline\_3$ AND $Baseline\_4$ ARE THE SAME. AND, ON WP, THE RESULTS OF $Baseline\_1$ AND $Baseline\_4$ ARE THE SAME. B_1, B_2, B_3 AND B_4 REPRESENT FOR BASELINE_1, BASELINE_2, BASELINE_3 AND BASELINE_4.

		B_1		B_2		B_3		B_4		DZN		QueryDet		Ours	
		CPD	WP	CPD	WP	CPD	WP	CPD	WP	CPD	WP	CPD	WP	CPD	WP
AP	**SSD**	0.448	0.264	0.415	0.355	0.448	0.243	0.448	0.264	0.504	0.382	0.507	0.258	**0.512**	**0.402**
	Yolov7	0.456	0.279	0.440	0.361	0.456	0.260	0.449	0.274	0.515	0.398	0.512	0.297	**0.518**	**0.424**
	FRCNN	0.451	0.277	0.438	0.359	0.451	0.256	0.451	0.277	**0.513**	0.394	0.509	0.304	0.510	**0.418**
P_{pn}		100%	100%	124%	124%	100%	16%	100%	100%	100%	48%	100%	44%	100%	46%
Inference time		0.033	0.128	0.297	1.152	0.033	0.048	0.033	0.272	0.097	0.338	0.082	0.305	0.086	0.316

TABLE II

PERFORMANCE COMPARISON ON THE VISDRONE-DET IN TERMS OF $AP \uparrow$, $P_{pn} \downarrow$ AND INFERENCE TIME↓, B_1, B_2, B_3 AND B_4 REPRESENT FOR BASELINE_1, BASELINE_2, BASELINE_3 AND BASELINE_4.

	B_1	B_2	B_3	B_4	DZN				QueryDet				Ours			
					AP	AP_S	AP_M	AP_L	AP	AP_S	AP_M	AP_L	AP	AP_S	AP_M	AP_L
SSD	0.226	0.271	0.205	0.254	0.285	0.198	0.411	0.526	0.284	0.192	0.408	0.535	**0.305**	0.208	0.417	0.544
Yolov7	0.239	0.291	0.226	0.257	0.295	0.205	0.420	0.527	0.289	0.199	0.415	0.544	**0.326**	0.225	0.436	0.550
FRCNN	0.234	0.280	0.212	0.259	0.288	0.210	0.416	0.514	0.279	0.204	0.410	0.537	**0.318**	0.233	0.424	0.542
P_{pn}	100%	125%	26%	100%	44%				41%				42%			
Inference time	0.041	0.492	0.054	0.306	0.114				0.096				0.096			

TABLE III

PERFORMANCE COMPARISON ON THE OGST IN TERMS OF $AP \uparrow$, $P_{pn} \downarrow$ AND INFERENCE TIME↓, B_1, B_2, B_3 AND B_4 REPRESENT FOR BASELINE_1, BASELINE_2, BASELINE_3 AND BASELINE_4.

		B_1	B_2	B_3	B_4	DZN	QueryDet	Ours
AP	**SSD**	0.759	0.838	0.733	0.802	0.846	**0.850**	0.848
	Yolov7	0.770	0.851	0.747	0.809	0.860	0.849	**0.863**
	FRCNN	0.763	0.843	0.740	0.814	0.844	0.832	**0.857**
P_{pn}		100%	137%	12.5%	100%	26%	28%	25%
Inference time		0.022	0.198	0.028	0.112	0.055	0.053	0.052

During inference, $Baseline\_1$ takes the shortest inference time. This is because its detector was trained on LR images, and during evaluation, HR images are directly resized to the LR dimensions. On the other hand, $Baseline\_2$ takes the longest inference time, as it needs to visit predefined regions n_a times to complete the evaluation of each HR image. $Baseline\_3$ and $Baseline\_4$ use detectors trained on HR images, which makes their inference time higher than that of $Baseline\_1$. While $Baseline\_3$ evaluates HR images in LR dimensions, $Baseline\_4$ performs the evaluation in the HR dimensions, and therefore, $Baseline\_4$ takes longer inference time than $Baseline\_3$. DZN, QueryDet, and our method all use detectors trained on LR images. However, their implementation is more complex than that of $Baseline\_1$, which results in a higher inference time than $Baseline\_1$ but lower than $Baseline\_3$ and $Baseline\_4$, which use detectors trained on HR images. Please note that the inference time on all datasets follows the same trend since the experimental setup is the same for all datasets.

B. VisDrone-Det

The evaluation results of different methods on VisDrone-Det are listed in Table II. Among the results of four Baselines, similar to the discoveries on CPD-WP, $Baseline\_2$ performs the best regarding AP and processes the most pixels (P_{pn} is 125%), and $Baseline\_3$ is the worst and processes the lowest percentage of pixels in HR images (P_{pn} is 26%). Out of the three CLQ-related methods, our method consistently outperforms the other two by at least 2% improvement on AP across different detectors. The improvement is more significant in comparison to the one on CPD. This is because (i) VisDrone-Det is more challenging and diverse; (ii) Since DZN uses two detectors, once the CLQ wrongly selects background regions as small object regions, the regions will be scanned by the fine detector (trained on HR images). This will significantly decrease the detection accuracy as reported in [47]; (iii) the CLQ in the QueryDet is learnt in the training stage. Hence, images in the testing set do not share exactly the same object scale as the ones in the training set, as the VisDrone-Det is very diverse. By comparing the values of *AP_S*, *AP_M*, and *AP_L* for each method, we can observe that the majority of the AP gain from each method is contributed by *AP_S*. Additionally, we can see that the values of *AP_S*, *AP_M*, and *AP_L* for the three CLQ related methods are comparable. Finally, for P_{pn}, there is no big difference among the three methods which process around 40% of the pixels in HR images. The trend of inference time among all methods on this dataset is the same as that on CPD-WP.

C. OGST

Table III shows the performance comparison of all methods on OGST. From the results, similar discoveries can be obtained among the four baselines, but the AP are much higher. The three CLQ-related methods not only demonstrate significantly better performance on OGST than on CPD-WP and VisDrone-Det, but also process much fewer pixels. This is because the location of objects in OGST is more concentrated compared with the other two datasets which can be also illustrated from the sample image in Fig.5. Thus, it is much easier for the CLQ to find the regions of small objects. Comparisons among the three methods show that our method can achieve comparable performance with the QueryDet and slightly better performance than the DZN. The trend of inference time among

TABLE IV
PERFORMANCE COMPARISON ON THE MS COCO MINI-VAL SET IN TERMS OF $AP \uparrow$, $P_{pn} \downarrow$ AND INFERENCE TIME↓, B_1 REPRESENTS FOR BASELINE_1.

	B_1				DZN				QueryDet				Ours			
	AP	AP_S	AP_M	AP_L	AP	AP_S	AP_M	AP_L	AP	AP_S	AP_M	AP_L	AP	AP_S	AP_M	AP_L
SSD	0.365	0.205	0.411	0.490	0.377	0.234	0.413	0.492	0.384	0.243	0.419	0.495	**0.385**	0.247	0.416	0.494
Yolov7	0.375	0.232	0.417	0.494	0.380	0.242	0.422	0.493	0.387	0.244	0.420	0.503	**0.390**	0.250	0.422	0.501
FRCNN	0.367	0.228	0.423	0.494	0.384	0.242	0.430	0.495	0.382	0.236	0.419	0.495	**0.385**	0.241	0.418	0.501
P_{pn}	100%				100%				100%				100%			
Inference time	0.033				0.095				0.075				0.088			

all methods on this dataset is verified to follow the same as that on CPD-WP and VisDrone-Det.

D. MS COCO mini-val

Table IV presents a comparison of evaluation results using different methods on the MS COCO mini-val set. As the experimental setup on this dataset is the same as on CPD, we can expect similar performance trends among Baseline_1 to Baseline_4 as on CPD. Therefore, we have chosen to only show the results of Baseline_1, which directly applies the detector on testing images since HR images are the same as LR images in this dataset. Comparing the results of Baseline_1 with CLQ related methods on *AP_S*, *AP_M*, and *AP_L* can help us determine whether the improvement in detection accuracy on small objects comes at the cost of detection accuracy on middle or large objects. After comparing the results, it is evident that our method improves the detection accuracy of small objects, while maintaining similar performance for medium and large objects, when compared to the baseline detectors, DZN and QueryDet.

TABLE V
THE NUMBER OF FLOPS (IN MILLION) FOR DZN, QUERYDET AND OUR METHOD (DECOMPOSED INTO STN, EVP AND TRANSFORMER)

	DZN	QueryDet	STN	EVP	Transformer
FLOPs	2.043M	1.797M	1.269M	0.579M	0.066M

After analyzing the results on four datasets, we can conclude that our method outperforms Baseline_1 to Baseline_4 as well as the other two CLQ-related methods (DZN and QueryDet) in terms of detection accuracy. To further demonstrate the computational efficiency of our method, we compared it with the other two CLQ-related methods in terms of model complexity . Specifically, we measured the number of FLOPs required by each method. The policy network of our model comprises three distinct components (the STN, EVP and Transformer) and we separately calculated the FLOPs for each component. It is worth noting that the number of FLOPs has a relation with output vector dimension, and thus varies across different datasets. To provide a representative estimate, we report the average number of FLOPs computed over four datasets for each component. The results are presented in Table V. Based on the total number of FLOPs, we can conclude that our method is computationally efficient, as it is comparable to the other two CLQ-related methods, despite having slightly higher FLOPs than QueryDet and lower FLOPs than DZN.

The visualization of detection results of sample images from VisDrone-Det and WP are shown in Fig. 7. The visualization of detection results on two samples from OGST are shown in Fig. 8. It can be seen from the first column in the figures that the CLQ in our method can locate the regions of small objects. The remaining columns in the figure indicate that the detector is able to detect most small objects from regions in HR.

TABLE VI
PERFORMANCE COMPARISON AMONG THREE VARIANTS OF OUR FRAMEWORK. FOR EACH VARIANT, THE FIRST AND SECOND COLUMN SHOW RESULTS ON VISDRONE-DET AND OGST RESPECTIVELY.

	W/O STN		W/O T		W/O CNN		Full Framework	
SSD	0.294	0.838	0.299	0.845	0.302	0.844	0.305	0.848
Yolov7	0.311	0.849	0.317	0.857	0.321	0.852	0.326	0.863
FRCNN	0.310	0.844	0.312	0.853	0.314	0.852	0.318	0.857

TABLE VII
AP COMPARISON USING OUR METHOD WITH YOLOV7 ON VISDRONE-DET BY VARYING THE SMALLEST OBJECT SIZE AND IOU THRESHOLD.

IoU \ Obj_size	20	30	40
0.30	0.471	0.563	0.668
0.50	0.339	0.465	0.597
0.75	0.207	0.282	0.376

E. Ablation Study

As shown in Fig.4, the policy network contains multiple components. We thus implement an ablation study to analyze the contribution of each component to performance improvement. Three variants of our framework are constructed: *W/O STN*: remove the STN component, *W/O CNN*: only use the patchify stem in EVP to generate input to the encoder of the Transformer backbone, and *W/O T*: remove the transform backbone and only use CNN to process feature map from output feature map of STN. The variants are evaluated on VisDrone-Det and OGST.

The evaluation results of the variants and the full framework are shown in Table VI. Comparing the results of each variant with the ones of the full framework, it can be observed that the performance drops after removing any of the components from our framework. However, the performance degradation is more significant on VisDrone-Det than OGST, especially when the STN is absent. This demonstrates that the STN plays a more important role in the performance improvement of the policy network on the challenging task.

Furthermore, the obtained average rewards of the three variants and the full framework in the RL agent training on VisDrone-Det are shown in Fig. 9. It is clear that the agents trained with the full policy network and W/O STN obtained the highest and lowest average reward after converging, re-

Fig. 7. Visualization of detection results from (a) the VisDrone-Det and (b) the WP. For each example, the first column shows the detection results (in dotted boxes) on LR image combined with the boundary of regions (in red boxes) selected by the CLQ. The remaining columns shows the detection results (in solid boxes) on selected regions in HR.

Fig. 8. Visualization of detection results in two samples from OGST. Red boxes in the first column indicate selected regions of small objects by the CLQ, and dotted green boxes show the coarsely detected objects from LR images. Detected small objects in the remaining columns are shown in solid green boxes.

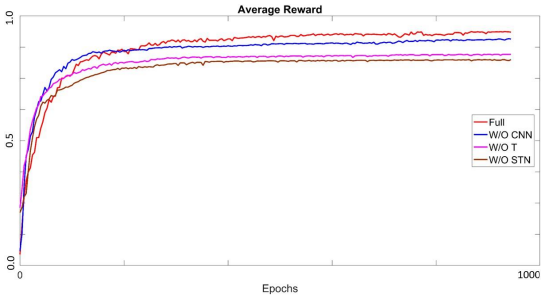

Fig. 9. The average reward obtained by training the RL agent with different variants of the proposed framework of the policy network.

spectively. This observation is in line with the results shown in Table VI.

We also conducted another ablation study to investigate the impact of the varying threshold of some factors on detection accuracy. Specifically, it's important to understand how adjusting the IoU threshold (in Section V-C) affects object detection, as well as how the detector's smallest object size can be customized to suit user requirements. To address these issues, we evaluated our method on VisDrone-Det using Yolov7 as the detector, where we varied the smallest object size from 20 to 40 pixels and the IoU threshold from 0.3 to 0.75. The evaluation results, presented in Table VII, demonstrate the impact of these variations on detection accuracy. We observed that increasing the smallest object size (i.e., making the detection task easier) resulted in improved detection accuracy. On the other hand, as the IoU threshold increased (i.e., the criteria for object detection became stricter), the accuracy decreased. It is important to note that variations in metric thresholds or user requirements can lead to different performance levels, which are not directly related to the method itself.

VII. Conclusions

In this paper, we propose a novel method to boost the accuracy of small object detection. Specifically, we employ an RL algorithm and a particularly designed policy network to train the agent (CLQ) to select regions of small objects. The performance of the CLQ relies on the novel policy network design by combining an STN, Transformer and CNN and the single-step multi-action training protocol. The proposed method is evaluated on four different datasets including street view images, UAV images, and remote sensing images. The experimental results show that our method can achieve comparable and better performance than the baselines and the SotA methods. In addition, the findings from the ablation study suggest that enhancing the state representation learning using an STN can significantly boost the CLQ accuracy; applying EVP (consisting of CNN and patchify stem) and Transformer in the policy network also play important roles for the accuracy improvement.

References

[1] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge", *International Journal of Computer Vision*, 88(2):303–338, 2010.

[2] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick," Microsoft coco: Common objects in context", *In European Conference on Computer Vision (ECCV)*, pages 740–755. Springer, 2014.

[3] M. Gao, R. Yu, A. Li, V.I. Morariu and L.S. Davis," Dynamic Zoom-in Network for East Object Detection in Large Images", *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6926-6935, 2018.

[4] M. Najibi, B. Singh and L.S. Davis, "AutoFocus: Efficient Multi-Scale Inference", *In International Conference on Computer Vision (ICCV)*, pages 9745-9755, 2018.

[5] C. Yang, Z. Huang and N. Wang, "QueryDet: Cascaded Sparse Query for Accelerating High-Resolution Small Object Detection", *In IEEE conference on computer vision and pattern recognition (CVPR)*, pages 13668-13677, 2022.

[6] Y. Kim, B-N. Kang and D. Kim, "SAN: Learning relationship between convolutional features for multi-scale object detection", *in European Conference on Computer Vision (ECCV)*, 2018.

[7] B. Singh and L.S. Davis, "An analysis of scale in-variance in object detection snip", *in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3578-3587, 2018.

[8] B. Singh, M. Najibi and L.S. Davis, "Sniper: Efficient multi-scale training", *In Advances in Neural Information Processing Systems (NIPS)*, pages 9310–9320, 2018.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.Y. Fu and C.B. Alexander, "SSD: Single Shot MultiBox Detector", *in European Conference on Computer Vision (ECCV)*, 2016.

[10] W. Ma, X. Wang and J. Yu, "A Lightweight Feature Fusion Single Shot Multibox Detector for Garbage Detection", *in IEEE Access*, vol. 8, pp. 188577-188586, 2020, doi: 10.1109/ACCESS.2020.3031990.

[11] X. Liang, J. Zhang, L. Zhuo, Y. Li and Q. Tian,"Small Object Detection in Unmanned Aerial Vehicle Images Using Feature Fusion and Scaling-Based Single Shot Detector with Spatial Context Analysis", *in IEEE Transactions on Circuits and Systems for Technology (TCSVT)*, vol. 30, No. 6, 2020.

[12] K. Cheng, H. Cui, H. A. Ghafoor, H. Wan, Q. Mao and Y. Zhan, "Tiny Object Detection via Regional Cross Self-Attention Network," *in IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, doi: 10.1109/TCSVT.2022.3232688.

[13] S. Liu, L. Qi, H. Qin, J. Shi and J. Jia,"Path aggregation network for instance segmentation", *in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8759-8768, 2018.

[14] M. Tan, R. Pang and Q.V. Le, "Efficientdet: Scalable and efficient object detection", *in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10781-10790, 2020.

[15] S. Qiao, L-C. Chen and A. Yuille, "Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution", *in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.10213-10224, 2021.

[16] Y. Li, Y. Chen, N. Wang and Z. Zhang, "Scale-aware trident networks for object detection", *in International Conference on Computer Vision (ICCV)*, 2019.

[17] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang and S.Z. Li, "S3fd: Single shot scale-invariant face detector", *in International Conference on Computer Vision (ICCV)*, pp. 192-201, 2017.

[18] K. Shuang, Z. Lyu, J. Loo and W. Zhang, "Scale-balanced loss for object detection", *in Pattern Recognition*, volume. 117, 107997, 2021.

[19] P. Zhou, B. Ni, C. Geng, J. Hu and Y. Xu, "Scale-Transferrable Object Detection", *in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.528-537, 2018.

[20] X. Yu, Y. Gong, N. Jiang, Q. Ye and Z. Han, "Scale Match for Tiny Person Detection", *in IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[21] C. Dong, CC. Loy, K. He and X. Tang, "Image Super-Resolution Using Deep Convolutional Networks", *in IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, pp. 295-307, 2016.

[22] J. Kim, J.K. Lee, K.M. Lee, "Accurate Image Super-Resolution Using Very Deep Convolutional Networks", *in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[23] K. Duan, D. Du, H. Qi and Q. Huang, "Detecting Small Objects Using a Channel-Aware Deconvolutional Network", *in IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, vol. 30, no. 6, pp. 1639-1652, 2020.

[24] S. Mathe, A. Pirinen, and C. Sminchisescu, " Reinforcement learning for visual object detection", *in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2894–2902, 2016.

[25] B. Uzkent and C. Yeh, "Efficient Object Detection in Large Images using Deep Reinforcement Learning", *in IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[26] F. Fang, Q.L. Xu, Y. Cheng, Y. Sun and J-H. Lim, " Image Understanding with Reinforcement Learning: Auto-tuning Image Attributes and Model Parameters for Object Detection and Segmentation", *in IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, vol. 32, no. 10, pp. 6671 - 6685, 2022.

[27] R. Bellman, "On the Theory of Dynamic Programming", *in Porc Natl Acad Sci USA*, vol. 38(8), pp. 716-719, 1952.

[28] M. Zhao, G. Cao, X. Huang and L. Yang, "Hybrid Transformer-CNN for Real Image Denoising", *in IEEE Signal Processing Letters*, vol. 29, pp. 1252-1256, 2022.

[29] Q. Xu and Y. Qian, "Bidirectional Transformer for Video Deblurring", *in IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, vol. 32, no. 12, pp. 8450-8461, 2022.

[30] J. Ding, N. Xue, G-S. Xia, X. Bai, W. Yang, M.Y. Yang et al., "Object Detection in Aerial Images: A Large-Scale Benchmark and Challenges", *in IEEE Transactions on Pattern Analysis and Machine Intelligence*, DOI:10.1109/TPAMI.2021.3117983, 2021.

[31] M. Jaderberg, K. Simonyan, A. Zisserman and K. Kavukcuoglu, "Spatial Transformer Network", *in Advances in Neural Information Processing Systems(NIPS)*, 2015.

[32] A. Dosovitskiy, L.Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. "An image is worth 16×16 words: Transformers for image recognition at scale", *in Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[33] B. Heo, s. Yun, D. Han, S. Chun, J. Choe, S.J. Oh, "Rethinking Spatial Dimensions of Vision Transformers", *in International Conference on Computer Vision (ICCV)*, 2021.

[34] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, B. Guo, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", *in International Conference on Computer Vision (ICCV)*, 2021.

[35] T. Xiao, M. Singh, E. Mintun, T. Darrell, P. Dollar and R. Girshick, "Early Convolutions Help Transformers See Better", *in Advances in Neural Information Processing Systems(NIPS)*, 2021.

[36] X. Ding, X. Zhang, Y. Zhou, J. Han, G. Ding and J. Sun, "Scaling Up Kernels to 31×31: Revisiting Large Kernel Design in CNNs", *in IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11963-11975, 2022.

[37] P. Dollar, C. Wojek, B. Schiele and P. Perona, "Pedestrian detection: An evaluation of the state of the art", *in IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4), pp. 743-761, 2012.

[38] S. Kalkowski, C. Schulze, A. Dengel and D. Borth, "Real-time analysis and visualization of the yfcc100m dataset", *in Proceedings of the Workshop on Community Organized Multi modal Mining: Opportunities for Novel Solutions*, pp. 25-30, ACM, 2015.

[39] D. Du, L. Wen, P. Zhu, H. Fan, Q. Hu, H. Ling et al., "VisDrone-DET2020: The Vision Meets Drone Object Detection in Image Challenge Results", *in European Conference on Computer Vision (ECCV) Workshop*, pp. 692-712, 2020.

[40] J. Rabbi and N. Ray and M. Schubert and S. Chowdhury and D. Chao, "Small-Object Detection in Rewmote Sensing Images with End-to-End Edge-Enhanced GAN and Object Detector Network", *in Remote Sensing*, 12, 1432, 2020.

[41] C-Y. Wang, A. Bochkovskiy and H-Y.M. Liao, "YOLOv7: Trainable bag-of-greebies sets new state-of-the-art for real-time object detectors", *in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[42] S.Q. Ren, K.M. He, R.G. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", *in Advances in Neural Information Processing Systems(NIPS)*, 2015.

[43] K. Diederik and B. Jimmy,"Adam: A Method for Stochastic Optimizatioin", *in International Conference on Learning Representations (ICLR)*, 2014.

[44] T. Schaul, J. Quan, I. Antonoglou and D. Silver, "Prioritized Experience Replay", *in International Conference on Learning Representations (ICLR)*, 2016.

[45] J. Lei, Q. Luan, X. Song, X. Liu, D. Tao and M. Song, "Action Parsing-Driven Video Summarization Based on Reinforcement Learning", *in IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, vol. 29, no. 7, pp. 2126-2137, 2019.

[46] R. Agarwal, "The Epsilon Greedy Algorithm - a Performance Review", *in International Journal of New Technology*, Vol. 6, No. 9, 2020, pp. 01-03.

[47] J. Rabbi, N. Ray, M. Schubert and S. Chowdhury, "Small-Object Detectiion in Remote Sensing Images with End-to-End Edge-Enhanced GAN and Object Detector Network", *in Remote Sensing*, vol. 12(9), pp. 1432, 2020.

[48] Z. Li, F. Nie, X. Chang, L. Nie, H. Zhang and Y. Yang, "Rank-Constrained Spectral Clustering With Flexible Embedding", *in IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 12, pp. 6073-6082, Dec. 2018, doi: 10.1109/TNNLS.2018.2817538.

[49] Z. Li, F. Nie, X. Chang, Y. Yang, C. Zhang and N. Sebe, "Dynamic Affinity Graph Construction for Spectral Clustering Using Multiple Features", *in IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 12, pp. 6323-6332, Dec. 2018, doi: 10.1109/TNNLS.2018.2829867.

[50] Z. Li, L. Yao, X. Chang, K. Zhan, J. Sun, H. Zhang, "Zero-shot event detection via event-adaptive concept relevance mining", *in Pattern Recognition*, Vol. 88, 2019, Pages 595-603.

[51] D. Zhang, J. Han, L. Zhao and T. Zhao,"From Discriminant to Complete: Reinforcement Searching-Agent Learning for Weakly Supervised Object Detection," *in IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 12, pp. 5549-5560, Dec. 2020, doi: 10.1109/TNNLS.2020.2969483.

[52] J. Han, L. Yang, D. Zhang, X. Chang and X. Liang,"Reinforcement Cutting-Agent Learning for Video Object Segmentation," *in IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9080-9089, 2018.

[53] F. Yang, H. Fan, P. Chu, E. Blasch and H. Ling,"Clustered Object Detection in Aerial Images," *in 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South), 2019, pp. 8310-8319.

[54] C. Li, T. Yang, S. Zhu, C. Chen and S. Guan,"Density Map Guided Object Detection in Aerial Images," *in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Seattle, WA, USA, 2020, pp. 737-746.

[55] F. Ö. Ünel, B. O. Özkalayci and C. Çiğla, "The Power of Tiling for Small Object Detection," *in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Long Beach, CA, USA, 2019, pp. 582-591.

Fen Fang received the Ph.D. degree in computer science and engineering, majoring in computer graphics, from Nanyang Technological University, Singapore, in 2014. She is currently a Research Scientist with the Institute for Infocomm Research, A*STAR, Singapore. Her research interests include defect detection, object detection and segmentation, scene recognition, and Reinforcement Learning.

Wenyu Liang (Member, IEEE) received the BEng and MEng degrees in mechanical engineering from China Agricultural University, Beijing, China, in 2008 and 2010, respectively, and the PhD degree in electrical and computer engineering from the National University of Singapore (NUS), Singapore, in 2014. He is currently a Scientist with the Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore, and also an Adjunct Assistant Professor with the Department of Electrical and Computer Engineering, NUS. His research interests include robotics, intelligent systems, dexterous manipulation, precision motion control, and force control.

Yi Cheng received the M.S. degree in Statistics from the National University of Singapore, Singapore, in 2017. She is currently a Senior Research Engineer with the Visual Intelligence Department, Institute for Infocomm Research (I2R), A*STAR, Singapore. Her research interest includes deep learning on computer vision, with an emphasis on video understanding and analysis.

Qianli Xu (M'07) received the B.E. and M.E. degrees from Tianjin University, Tianjin, China, in 1999 and 2002, respectively, and the Ph.D. degree from the National University of Singapore, Singapore, in 2007. He is currently a senior scientist with the Visual Intelligence Department, Institute for Infocomm Research, Singapore. His current research interests include computer vision, human–computer interaction, health informatics, and interaction design. His publications appear in NeurIPS, AAAI, ACM Multimedia, IEEE ICRA, ACM SIGCHI, HRI, Scientific Reports, the IEEE Transactions on Systems, Man, and Cybernetics: Systems, the IEEE Transactions on Cybernetics, the IEEE Journal of Biomedical and Health Informatics, etc.

Joo-Hwee Lim (SM'17) received the B.Sc. (Hons.I) and M.Sc. (research) degrees in computer science from NUS and the Ph.D. degree in computer science and engineering from UNSW. He has published over 280 international refereed journals and conference papers and has coauthored 25 patents (awarded and pending). He is currently a Principal Scientist and the Head of the Visual Intelligence Department, Institute of Infocomm Research, A-STAR, Singapore.