HoughNet: Integrating near and long-range evidence for bottom-up object detection

Nermin Samet[1], Samet Hicsonmez[2], and Emre Akbas[1]

[1] Department of Computer Engineering, Middle East Technical University
{nermin,emre}@ceng.metu.edu.tr
[2] Department of Computer Engineering, Hacettepe University
{samethicsonmez}@hacettepe.edu.tr

Abstract. This paper presents HoughNet, a one-stage, anchor-free, voting-based, bottom-up object detection method. Inspired by the Generalized Hough Transform, HoughNet determines the presence of an object at a certain location by the sum of the votes cast on that location. Votes are collected from both near and long-distance locations based on a log-polar vote field. Thanks to this voting mechanism, HoughNet is able to integrate both near and long-range, class-conditional evidence for visual recognition, thereby generalizing and enhancing current object detection methodology, which typically relies on only local evidence. On the COCO dataset, HoughNet's best model achieves 46.4 AP (and 65.1 AP_{50}), performing on par with the state-of-the-art in bottom-up object detection and outperforming most major one-stage and two-stage methods. We further validate the effectiveness of our proposal in another task, namely, "labels to photo" image generation by integrating the voting module of HoughNet to two different GAN models and showing that the accuracy is significantly improved in both cases. Code is available at https://github.com/nerminsamet/houghnet.

Keywords: Object Detection, Voting, Bottom-up recognition, Hough Transform, Image-to-image translation

1 Introduction

Deep learning has brought on remarkable improvements in object detection. Performance on widely used benchmark datasets, as measured by mean average-precision (mAP), has at least doubled (from 0.33 mAP [15] [11] to 0.80 mAP on PASCAL VOC [17]; and from 0.2 mAP [28] to around 0.5 mAP on COCO [27]) in comparison to the previous generation (pre-deep-learning, shallow) methods. Current state-of-the-art, deep learning based object detectors [27,30,38,41] predominantly follow a top-down approach where objects are detected holistically via rectangular region classification. This was not the case with the pre-deep-learning methods. The bottom-up approach was a major research focus as exemplified by the prominent voting-based (the Implicit Shape Model [24]) and part-based (the Deformable Parts Model [10]) methods. However, today, among

deep learning based object detectors, the bottom-up approach has not been sufficiently explored with a few exceptions (e.g. CornerNet [23], ExtremeNet [52]).

In this paper, we propose Hough-Net, a one-stage, anchor-free, voting-based, bottom-up object detection method. HoughNet is based on the idea of voting, inspired by the Generalized Hough Transform [2, 18]. In its most generic form, the goal of GHT is to detect a whole shape based on its parts. Each part produces a hypothesis, i.e. casts its vote, regarding the location of the whole shape. Then, the location with the most votes is selected as the result. Similarly, in HoughNet, the presence of an object belonging to a certain class at a particular location is determined by the sum of the class-conditional votes cast on that location (Fig. 1). HoughNet

Fig. 1: (Left) A sample "mouse" detection, shown with yellow bounding box, by HoughNet. (Right) The locations that vote for this detection. Colors indicate vote strength. In addition to the local votes originating from the mouse itself, there are strong votes from nearby "keyboard" objects, which shows that HoughNet is able to utilize both short and long-range evidence for detection. More examples can be seen in Fig. 4

processes the input image using a convolutional neural network to produce an intermediate score map per class. Scores in these maps indicate the presence of visual structures that would support the detection of an object instance. These structures could be object parts, partial objects or patterns belonging to the same or other classes. We name these score maps as "visual evidence" maps. Each spatial location in a visual evidence map votes for target areas that are likely to contain objects. Target areas are determined by placing a log-polar grid, which we call the "vote field," centered at the voter location. The purpose of using a log-polar vote field is to reduce the spatial precision of the vote as the distance between voter location and target area increases. This is inspired by foveated vision systems found in nature, where the spatial resolution rapidly decreases from the fovea towards the periphery [22]. Once all visual evidence is processed through voting, the accumulated votes are recorded in object presence maps, where the peaks indicate the presence of object instances.

Current state-of-the-art object detectors rely on local (or short-range) visual evidence to decide whether there is an object at that location (as in top-down methods) or an important keypoint such as a corner (as in bottom-up methods). On the other hand, HoughNet is able to integrate both short and long-range visual evidence through voting. An example is illustrated in Fig. 1, where the detected mouse gets strong votes from two keyboards, one of which is literally at the other side of the image. In another example (Fig. 4, row 2, col 1), a ball on the right-edge of the image is voting for the baseball bat on the left-edge. On the COCO dataset, HoughNet achieves comparable results with the state-of-the-art bottom-up detector CenterNet [9], while being the fastest object detector among bottom-up detectors. It outperforms prominent one-stage (RetinaNet [27]) and

two-stage detectors (Faster RCNN [41], Mask RCNN [16]). To further show the effectiveness of our approach, we used the voting module of HoughNet in another task, namely, "labels to photo" image generation. Specifically, we integrated the voting module to two different GAN models (CycleGAN [54] and Pix2Pix [20]) and showed that the performance is improved in both cases.

Our main contribution in this work is HoughNet, a voting-based bottom-up object detection method that is able to integrate near and long-range evidence for object detection. As a minor contribution, we created a mini training set called "COCO minitrain", a curated subset of COCO train2017 set, to reduce the computational cost of ablation experiments. We validated COCO minitrain in two ways by (i) showing that the COCO val2017 performance of a model trained on COCO minitrain is strongly positively correlated with the performance of the same model trained on COCO train2017, and (ii) showing that COCO minitrain set preserves the object instance statististics.

2   Related Work

Methods using log-polar fields/representations. Many biological systems have foveated vision where the spatial resolution decreases from the fovea (point of fixation) towards the periphery. Inspired by this phenomenon, computer vision researchers have used log-polar fields for many different purposes including shape description [4], feature extraction [1] and foveated sampling/imaging [44].

Non-deep, voting-based object detection methods. In the pre-deep learning era, generalized Hough Transform (GHT) based voting methods have been used for object detection. The most influential work was the Implicit Shape Model (ISM) [24]. In ISM, Leibe et al. [24] applied GHT for object detection/recognition and segmentation. During the training of the ISM, first, interest points are extracted and then a visual codebook (i.e. dictionary) is created using an unsupervised clustering algorithm applied on the patches extracted around interest points. Next, the algorithm matches the patches around each interest point to the visual word with the smallest distance. In the last step, the positions of the patches relative to the center of the object are associated with the corresponding visual words and stored in a table. During inference, patches extracted around interest points are matched to closest visual words. Each matched visual word casts votes for the object center. In the last stage, the location that has the most votes is identified, and object detection is performed using the patches that vote for this location. Later, ISM was further extended with discriminative frameworks [3, 14, 32, 33, 37]. Okada [33] ensembled randomized trees using image patches as voting elements. Similarly, Gall and Lempitsky [14] proposed to learn a mapping between image patches and votes using random forest framework. In order to fix the accumulation of inconsistent votes of ISM, Razavi et al. [37] augmented the Hough space with latent variables to enforce consistency between votes. In Max-margin Hough Transform [32], Maji and Malik showed

the importance of learning visual words in a discriminative max-margin framework. Barinova et al. [3] detected multiple objects using energy optimization instead of non-maxima suppression peak selection of ISM.

HoughNet is similar to ISM and its variants described above only at the idea level as all are voting based methods. There are two major differences: (i) HoughNet uses deep neural networks for part/feature (i.e. visual evidence) estimation, whereas ISM uses hand-crafted features; (ii) ISM uses a discrete set of visual words (obtained by unsupervised clustering) and each word's vote is exactly known (stored in a table) after training. In HoughNet, however, there is not a discrete set of words and vote is carried through a log-polar vote field which takes into account the location precision as a function of target area.

Bottom-up object detection methods. Apart from the classical one-stage [12, 27, 30, 39, 40] vs. two-stage [16, 41] categorization of object detectors, we can also categorize the current approaches into two: top-down and bottom-up. In the top-down approach [27, 30, 39, 41], a near-exhaustive list of object hypotheses in the form of rectangular boxes are generated and objects are predicted in a holistic manner based on these boxes. Designing the hypotheses space (e.g. parameters of anchor boxes) is a problem by itself [45]. Typically, a single template is responsible for the detection of the whole object. In this sense, recent anchor-free methods [43, 51] are also top-down. On the other hand, in the bottom-up approach, objects *emerge* from the detection of parts (or sub-object structures). For example, in CornerNet [23], top-left and bottom-right corners of objects are detected first, and then, they are paired to form whole objects. Following CornerNet, ExtremeNet [52] groups extreme points (e.g. left-most, etc.) and center points to form objects. Together with corner pairs of CornerNet [23], CenterNet [9] adds center point to model each object as a triplet. HoughNet follows the bottom-up approach based on a voting strategy: object presence score is voted (aggregated) from a wide area covering short and long-range evidence.

Deep, voting-based object detection methods. Qi et al. [36] apply Hough voting for 3D object detection in point clouds. Sheshkus et al. [42] utilize Hough transform for vanishing points detection in the documents. For automatic pedestrian and car detection, Gabriel et al. [13] proposed using discriminative generalized Hough transform for proposal generation in edge images, later to further refine the boxes, they fed these proposals to deep networks. In the deep learning era, we are not the first to use a log-polar vote field in a voting-based model. Lifshitz et al. [25] used a log-polar map to estimate keypoints for single person human pose estimation. Apart from the fact that [25] is tackling the human pose estimation task, there are several subtle differences. First, they prepare ground truth voting maps for each keypoint such that keypoints vote for every other one depending on its relative position in the log polar map. This requires manually creating static voting maps. Specifically, their model learns $H \times W \times R \times C$ voting map, where R is the number of bins and C is the augmented keypoints. In order to produce keypoint heatmaps they perform vote agregation at test phase. Second, this design restricts the model to learn only the keypoint locations as
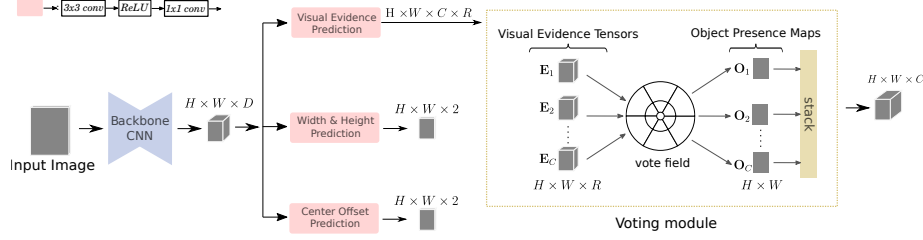
Fig. 2: Overall processing pipeline of HoughNet

voters. When we consider the object detection task and its complexity, it is not trivial to decide the voters of the objects and prepare supervised static voting maps as in human pose estimation. Moreover, this design limits the voters to reside only inside of the object (e.g. person) unlike our approach where an object could get votes from far away regions. To overcome these issues, unlike their model we apply vote aggregation during training (they perform vote agregation only at test phase). This allows us to expose the latent patterns between objects and voters for each class. In this way, our voting module is able to get votes from non-labeled objects (see the last row of Fig. 4). To the best of our knowledge, we are the first to use a log-polar vote field in a voting-based deep learning model to integrate the **long range interactions** for object detection.

Similar to HoughNet, Non-local neural networks (NLNN) [46] and Relation networks (RN) [19] integrate long-range features. As a fundamental difference, in NLNN, the relative displacement between interacting features is not taken into account. However, HoughNet uses this information encoded through the regions of the log-polar vote field. RN models object-object relations explicitly for proposal-based two-stage detectors.

3   HoughNet: the method and the models

The overall processing pipeline of our method is illustrated in Fig. 2. To give a brief overview, the input image first passes through a backbone CNN, the output of which is connected to three different branches carrying out the predictions of (i) visual evidence scores, (ii) objects' bounding box dimensions (width and height), and (iii) objects' center location offsets. The first branch is where the voting occurs. Before we describe our voting mechanism in detail, we first introduce the log-polar vote field.

3.1   The log-polar "vote field"

We use the set of regions in a standard log-polar coordinate system to define the regions through which votes are collected. A log-polar coordinate system is defined by the number and radii of eccentricity bins (or rings) and the number of angle bins. We call the set of cells or regions formed in such a coordinate

system as the "vote field" (Fig. 3). In our experiments, we used different vote fields with different parameters (number of angle bins, etc.) as explained in the Experiments section. In the following, R denotes the number of regions in the vote field and K_r is the number of pixels in a particular region r. $\Delta_r(i)$ denotes the relative spatial coordinates of the ith pixel in the rth region, with respect to the center of the field. We implement the vote field as a fixed-weight (non-learnable) transposed-convolution filter as further explained below.

3.2   Voting module

After the input image is passed through the backbone network and the "visual evidence" branch, the voting module of HoughNet receives C tensors $\mathbf{E}_1, \mathbf{E}_2, \ldots, \mathbf{E}_C$, each of size $H \times W \times R$, where C is the number of classes, H and W are spatial dimensions and R is the number of regions in the vote field. Each of these tensors contains class-conditional (i.e. for a specific class) "visual evidence" scores. The job of the voting module is to produce C "object presence" maps $\mathbf{O}_1, \mathbf{O}_2, \ldots, \mathbf{O}_C$, each of size $H \times W$. Then, peaks in these maps will indicate the presence of object instances. The voting process, which converts the visual evidence tensors (e.g. $\mathbf{E}_c$) to object presence maps (e.g. $\mathbf{O}_c$), works as described below.

Suppose we wanted to process the visual evidence at the ith row, j^{th} column and the rth channel of an evidence tensor $\mathbf{E}$. When we place our vote field on a 2D map, centered at location (i, j), the region r marks the target area to be voted on, whose coordinates can be

Fig. 3: A log-polar "vote field" used in the voting module of HoughNet. Numbers indicate region ids. A vote field is parametrized by the number of angle bins, and the number and radii of eccentricity bins, or rings. In this particular vote field, there are a total of 13 regions, 6 angle bins and 3 rings. The radii of the rings are 2, 8 and 16, respectively

computed by adding the coordinate offsets $\Delta_r(\cdot)$ to (i, j). Then, we add the visual evidence score $\mathbf{E}(i, j, r)$ to the target area of the object presence map. Note that this operation can be efficiently implemented using the "transposed convolution" (or "deconvolution") operation. Visual evidence scores from locations other than (i, j) are processed in the same way and the scores are accumulated in the object presence map. We formally define this procedure in Algorithm 1, which takes in a visual evidence tensor as input and produces an object presence map[3].

3.3   Network architecture

Our network architecture design follows that of "Objects as Points" (OAP) [51]. HoughNet consists of a backbone and three subsequent branches which predict
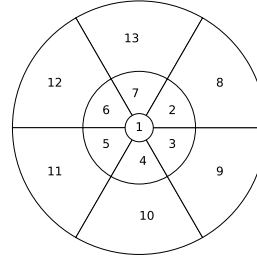
---

[3] We   provide   a   step-by-step   animation   of   the   voting   process   at   https://shorturl.at/ilOP2.

Algorithm 1 Vote aggregation algorithm

Input: Visual evidence tensor $\mathbf{E}_c$, Vote field relative coordinates Δ
Output: Object presence map $\mathbf{O}_c$
  Initialize $\mathbf{O}_c$ with all zeros
  for each pixel (i, j, r) in $\mathbf{E}_c$ **do**
    /* K_r: number of pixels in the vote field region r */
    for $k = 1$ to K_r **do**
      $(y, x) \leftarrow (i, j) + \Delta_r(k)$
      $\mathbf{O}_c(y, x) \leftarrow \mathbf{O}_c(y, x) + \frac{1}{K_r}\mathbf{E}_c(i, j, r)$
    end for
  end for

(i) visual evidence scores, (ii) bounding box widths and heights, and (iii) center offsets. Our voting module is attached to the visual evidence branch (Fig. 2).

The output of our backbone network is a feature map of size $H \times W \times D$, which is a result of inputting an image of size $4H \times 4W \times 3$. The backbone's output is fed to all three branches. Each branch has one convolutional layer with 3×3 filters followed by a ReLU layer and another convolutional layer with 1×1 filters. The visual evidence branch outputs $H \times W \times C \times R$ sized output where C and R correspond to the number of classes and vote field regions, respectively. The width/height prediction branch outputs $H \times W \times 2$ sized output which predicts heights and widths for each possible object center. Finally, center offset branch predicts relative displacement of center locations across the spatial axes.

Objective functions For the optimization of the visual evidence branch, we use the modified focal loss [27] introduced in CornerNet [23] (also used in [51,52]). In order to recover the lost precision of the center points due to down-sampling operations through the network, center offset prediction branch outputs class-agnostic local offsets of object centers. We optimize this branch using the L_1 loss as the other bottom-up detectors [23,51,52] do. Finally, our width & height prediction branch outputs class-agnostic width and height values of objects. For the optimization of this branch, we use L_1 loss by scaling the loss by 0.1 as proposed in OAP [51]. The overall loss is the sum of the losses from all branches.

4   Experiments

This section presents the experiments we conducted to show the effectiveness of our proposed method. First, we studied how different parameters of the vote field affect the final object detection performance. Next, we present several performance comparisons between HoughNet and the current state-of-the-art methods, on the COCO dataset. After presenting sample visual results for qualitative inspection, we describe our experiments on the "labels to photo" task. We used PyTorch [35] to implement HoughNet.

Training and inference details We ran our experiments on 4 V100 GPUs. For training, we used 512×512 images unless stated otherwise. The training setup is not uniform across different experiments, mainly due to different backbones.

The inference pipeline is common for all HoughNet models. We extract center locations by applying a 3×3 max pooling operation on object presence heatmaps and pick the highest scoring 100 points as detections. Then, we adjust these points using the predicted center offset values. Final bounding boxes are generated using the predicted width & height values on these detections. For testing, we follow the other bottom-up methods [23, 51, 52] and use two modes: (i) single-scale, horizontal-flip testing (SS testing mode), and (ii) multi-scale, horizontal-flip testing (MS testing mode). In MS, we use the following scale values, $0.6, 1.0, 1.2, 1.5, 1.8$. To merge augmented test results, we use Soft-NMS [5], and keep the top 100 detections. All tests are performed on a single V100 GPU.

4.1   Mini COCO

For faster analysis in our ablation experiments, we created "COCO `minitrain`" as a statistically validated mini training set. It is a subset of the COCO `train2017` dataset, containing $25K$ images (about 20% of `train2017`) and around $184K$ objects across 80 object categories. We randomly sampled these images from the full set while preserving the following three quantities as much as possible: (i) proportion of object instances from each class, (ii) overall ratios of small, medium and large objects, (iii) per class ratios of small, medium and large objects.

To validate COCO `minitrain`, we computed the correlation between the `val2017` performance of a model when it is trained on `minitrain` with the same of when it is trained on `train2017`. Over six different object detectors (Faster R-CNN, Mask R-CNN, RetinaNet, CornerNet, ExtremeNet and HoughNet), the Pearson correlation coefficients turned out to be 0.74 and 0.92 for AP and AP_{50}, respectively. These values indicate strong positive correlation. Further details on `minitrain` can be found at https://github.com/giddyyupp/coco-minitrain.

4.2   Ablation experiments

Here we analyze the effects of the number of angle and ring bins of the vote field on performance. Models are trained on COCO `minitrain` and evaluated on `val2017` set with SS testing mode. The backbone is Resnet-101 [17]. In order to get higher resolution feature maps, we add three deconvolution layers on top of the default Resnet-101 network, similar to [47]. We add 3×3 convolution filters before each 4×4 deconvolution layer, and put batchnorm and ReLU layers after convolution and deconvolution filters. We trained the network with a batch size of 44 for 140 epochs with Adam optimizer [21]. Initial learning rate 1.75×10^{-4} was divided by 10 at epochs 90 and 120.

Angle bins We started with a large, 65 by 65, vote field with 5 rings. We set the radius of these rings from the most inner one to the most outer one as 2, 8, 16, 32 and 64 pixels, respectively. We experimented with $60°$, $90°$, $180°$ and $360°$

bins. We do not split the center ring (i.e. region with id 1 in Fig. 3) into further regions. Results are presented in Table 1a. For the 180° experiment, we divide the vote field horizontally. 90° yields the best performance considering both AP and AP_{50}. We used this setting in the rest of the experiments.

Effects of center and periphery We conducted experiments to analyze the importance of votes coming from different rings of the vote field. Results are presented in Table 1b. In the *Only Center* case, we only keep the center ring and disable the rest. In this way, we only aggregate votes from features of the object center directly. This case corresponds to a traditional object detection paradigm where only local (short-range) evidence is used. This experiment shows that votes from outer rings help improve performance. For the *No Center* case, we only disable the center ring. We observe that there is only 0.2 decrease in AP. This suggests that the evidence for successful detection is embedded mostly around the object center not directly inside the object center. In order to observe the power of long-range votes, we conducted another experiment called "Only Context," where we disabled the two most inner rings and used only the three outer rings for vote aggregation. This model reduced AP by 1.0 point compared to the full model.

Table 1: Ablation experiments for the vote field. (a) Effect of angle bins on performance. Vote field with 90° has the best performance (considering AP and AP_{50}). (b) Effect of central and peripheral regions. Here, the angle bin is 90° and the ring count is four. Disabling any of center or periphery hurts performance, cf. (a). (c) Effect of number of rings. Angle is 90° and vote field size is updated according to the radius of the last ring. Using 3 rings yields the best result. It is also the fastest model

Model	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	FPS
60°	**24.6**	41.3	25.0	**8.2**	27.7	36.2	3.4
90°	**24.6**	**41.5**	25.0	**8.2**	27.7	36.2	**3.5**
180°	24.5	41.1	24.8	8.1	27.7	**36.3**	3.5
360°	**24.6**	41.1	**25.1**	8.0	**27.8**	**36.3**	**3.5**

(a) Varying the number of angle bins

Only Center	23.8	39.5	24.5	**7.9**	26.8	34.7	**3.5**
No Center	**24.4**	**40.9**	**24.9**	7.4	**27.6**	**37.1**	3.3
Only Context	23.6	39.7	24.2	7.4	26.4	35.9	3.4

(b) Effectiveness of votes from center or periphery

5 Rings	24.6	**41.5**	25.0	8.2	27.7	36.2	3.5
4 Rings	24.5	41.1	25.3	8.2	**27.8**	36.1	7.8
3 Rings	**24.8**	41.3	**25.6**	**8.4**	27.6	**37.5**	**15.6**

(c) Varying ring counts

Ring count To find out how far an object should get votes from, we discard outer ring layers one by one as presented in Table 1c. The models with 5 rings, 4 rings and 3 rings have 17, 13 and 9 voting regions and 65, 33 and 17 vote field sizes, respectively. The model with 3 rings yields the best performance on AP metric and is the fastest one at the same time. On the other hand, the model with 5 rings yields 0.2 AP_{50} improvement over the model with 3 rings.

From all these ablation experiments, we decided to use the model with 5 rings and 90° as our *Base Model*. Considering both speed and accuracy, we decided to use the model with 3 rings and 90° as our *Light Model*.

Voting module vs. dilated convolution Dilated convolution [48], which can include long-range features, could be considered as an alternative to our

voting module. To compare performance, we trained models on `train2017` and evaluated them on `val2017` using the SS testing mode.

Baseline: We consider OAP with ResNet-101-DCN backbone as baseline. The last 1×1 convolution layer of center prediction branch in OAP, receives $H \times W \times D$ tensor and outputs object center heatmaps with a tensor of size $H \times W \times C$.

Baseline + Voting Module: We first adapt the last layer of center prediction branch in baseline to output $H \times W \times C \times R$ tensor, then attach our voting module on top of the cen-

Table 2: Comparing our voting module to an equivalent (in terms of number of parameters and the spatial filter size) dilated convolution filter on COCO `val2017` set. Models are trained on COCO `train2017` and results are presented on SS testing mode

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Baseline	36.2	54.8	38.7	16.3	41.6	52.3
+ Dilated Conv.	36.6	56.1	39.2	16.7	42.0	53.6
+ Voting Module	**37.3**	**56.6**	**39.9**	**16.8**	**42.6**	**55.2**

ter prediction branch. Adding the voting module increases parameters of the layer by R times. The log-polar vote field is 65×65, and has 5 rings (90°). With 5 rings and 90° we end up with $R = 17$ regions.

Baseline + Dilated Convolution: We use dilated convolution with kernel size 4×4 and dilation rate 22 for the last layer of the center prediction branch in baseline. Using 4×4 kernel increases parameters 16 times which is approximately equal to R in the *Baseline + Voting Module*. Using dilation rate 22, the filter size becomes 67×67 which is close to 65×65 log-polar vote field.

For a fair comparison with *Baseline*, both *Baseline + Voting Module* and the *Baseline + Dilated Convolution* use Resnet-101-DCN backbone. Our voting module outperforms dilated convolution in all cases (Table 2).

Table 3: HoughNet results on COCO `val2017` set for different training setups. [†] indicates initialization with CornerNet weights, [*] indicates initialization with ExtremeNet weights. Results are given for SS and MS testing modes, respectively

Models	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	FPS
Base	R-101	36.0 / 40.7	55.2 / 60.6	38.4 / 43.9	16.2 / 22.5	41.7 / 44.2	52.0 / 55.7	3.5 / 0.5
Base	R-101-DCN	37.3 / 41.6	56.6 / 61.2	39.9 / 44.9	16.8 / 22.6	42.6 / 44.8	55.2 / 58.8	3.3 / 0.4
Light	R-101-DCN	37.2 / 41.5	56.5 / 61.5	39.6 / 44.5	16.8 / 22.5	42.5 / 44.8	54.9 / 58.4	**14.3** / 2.1
Light	HG-104	40.9 / 43.7	59.2 / 61.9	44.1 / 47.3	23.8 / 27.5	45.3 / 45.9	52.6 / 56.2	6.1 / 0.8
Light	HG-104*	41.7 / 44.7	60.5 / 63.2	45.6 / 48.9	23.9 / 28.0	45.7 / 47.0	54.6 / 58.1	5.9 / 0.8
Light	HG-104*	43.0 / **46.1**	62.2 / **64.6**	46.9 / **50.3**	25.5 / **30.0**	47.6 / **48.8**	55.8 / **59.7**	5.7 / 0.8

4.3   Performance of HoughNet and comparison with baseline

In Table 3, we present the performance of HoughNet for different backbone networks, initializations and our base-vs-light model, on the `val2017` set. There is a significant speed difference between Base and Light models. Our light model with R-101-DCN backbone is the fastest one (14.3 FPS) achieving 37.2 AP and 56.5 AP_{50}. We observe that initializing the backbone with a pretrained model improves the detection performance. In Table 4, we compare HoughNet's performance with its baseline OAP [51] for two different backbones. HoughNet is especially effective for small objects, it improves the baseline by 2.1 and 2.2

AP points for R-101-DCN and HG-104 backbones, respectively. We also provide results for the recently introduced *moLRP* [34] metric, which combines localization, precision and recall in a single metric. Lower values are better.

Table 4: Comparison with baseline (OAP) on `val2017`. Results are given for single scale and multi scale test modes, respectively

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	$moLRP \downarrow$
Baseline *w* R-101-DCN	36.2 / 39.2	54.8 / 58.6	38.7 / 41.9	16.3 / 20.5	41.6 / 42.6	52.3 / 56.2	71.1 / 68.3
+ Voting Module	**37.2 / 41.5**	**56.5 / 61.5**	**39.6 / 44.5**	**16.8 / 22.5**	**42.5 / 44.8**	**54.9 / 58.4**	**69.9 / 66.6**
Baseline *w* HG-104	42.2 / 45.1	61.1 / 63.5	46.0 / 49.3	25.2 / 27.8	46.4 / 47.7	55.2 / **60.3**	66.1 / 63.9
+ Voting Module	**43.0 / 46.1**	**62.2 / 64.6**	**46.9 / 50.3**	**25.5 / 30.0**	**47.6 / 48.8**	**55.8** / 59.7	**65.6 / 63.1**

Table 5: Comparison with the state-of-the-art on COCO `test-dev`. The methods are divided into three groups: two-stage, one-stage top-down and one-st age bottom-up. The best results are boldfaced separately for each group. Backbone names are shortened: R is ResNet, X is ResNeXt, F is FPN and HG is HourGlass. * indicates that the FPS values were obtained on the same AWS machine with a V100 GPU using the official repos in SS setup. The rest of the FPS are from their corresponding papers. F. R-CNN is Faster R-CNN

Method	Backbone	Initialize	Train size	Test size	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	FPS
Two-stage detectors:											
R-FCN [8]	R-101	ImageNet	800×800	600×600	29.9	51.9	–	10.8	32.8	45.0	5.9
CoupleNet [55]	R-101	ImageNet	ori.	ori.	34.4	54.8	37.2	13.4	38.1	50.8	–
F. R-CNN+++ [17]	R-101	ImageNet	1000×600	1000×600	34.9	55.7	37.4	15.6	38.7	50.9	–
F. R-CNN [26]	R-101-F	ImageNet	1000×600	1000×600	36.2	59.1	39.0	18.2	39.0	48.2	5.0
Mask R-CNN [16]	X-101-F	ImageNet	1300×800	1300×800	39.8	62.3	43.4	22.1	43.2	51.2	11.0
Cascade R-CNN [6]	R-101	ImageNet	-	-	42.8	62.1	46.3	23.7	45.5	55.2	**12.0**
PANet [29]	X-101	ImageNet	1400×840	1400×840	**47.4**	**67.2**	**51.8**	**30.1**	**51.7**	**60.0**	–
One-stage detectors:											
Top Down:											
SSD [30]	VGG-16	ImageNet	512×512	512×512	28.8	48.5	30.3	10.9	31.8	43.5	–
YOLOv3 [39]	Darknet	ImageNet	608×608	608×608	33.0	57.9	34.4	18.3	35.4	41.9	**20.0**
DSSD513 [12]	R-101	ImageNet	513×513	513×513	33.2	53.3	35.2	13.0	35.4	51.1	–
RefineDet (SS) [49]	R-101	ImageNet	512×512	512×512	36.4	57.5	39.5	16.6	39.9	51.4	–
RetinaNet [27]	X-101-F	ImageNet	1300×800	1300×800	40.8	61.1	44.1	24.1	44.2	51.2	5.4
RefineDet (MS) [49]	R-101	ImageNet	512×512	≤2.25×	41.8	62.9	45.7	25.6	45.1	54.1	–
OAP (SS) [51]	HG-104	ExtremeNet	512×512	ori.	42.1	61.1	45.9	24.1	45.5	52.8	9.6*
FSAF (SS) [53]	X-101	ImageNet	1300×800	1300×800	42.9	63.8	46.3	26.6	46.2	52.7	2.7
FSAF (MS) [53]	X-101	ImageNet	1300×800	∼≤2.0×	44.6	65.2	48.6	29.7	47.1	54.6	–
FCOS [43]	X-101-F	ImageNet	1300×800	1300×800	44.7	64.1	48.4	27.6	47.5	55.6	7.0*
FreeAnchor (SS) [50]	X-101-F	ImageNet	1300×960	1300×960	44.9	64.3	48.5	26.8	48.3	55.9	–
OAP (MS) [51]	HG-104	ExtremeNet	512×512	≤1.5×	45.1	63.9	49.3	26.6	47.1	57.7	–
FreeAnchor (MS) [50]	X-101-F	ImageNet	1300×960	∼≤2.0×	**47.3**	**66.3**	**51.5**	**30.6**	**50.4**	**59.0**	–
Bottom Up:											
ExtremeNet (SS) [52]	HG-104	-	511×511	ori.	40.2	55.5	43.2	20.4	43.2	53.1	3.0*
CornerNet (SS) [23]	HG-104	-	511×511	ori.	40.5	56.5	43.1	19.4	42.7	53.9	5.2*
CornerNet (MS) [23]	HG-104	-	511×511	≤1.5×	42.1	57.8	45.3	20.8	44.8	56.7	–
ExtremeNet (MS) [52]	HG-104	-	511×511	≤1.5×	43.7	60.5	47.0	24.1	46.9	57.6	–
CenterNet (SS) [9]	HG-104	-	511×511	ori.	44.9	62.4	48.1	25.6	47.4	57.4	4.8*
CenterNet (MS) [9]	HG-104	-	511×511	≤1.8×	**47.0**	64.5	**50.7**	28.9	**49.9**	**58.9**	–
HoughNet (SS)	HG-104	-	512×512	ori.	40.8	59.1	44.2	22.9	44.4	51.1	**6.4***
HoughNet (MS)	HG-104	-	512×512	≤1.8×	44.0	62.4	47.7	26.4	45.4	55.2	–
HoughNet (SS)	HG-104	ExtremeNet	512×512	ori.	43.1	62.2	46.8	24.6	47.0	54.4	**6.4***
HoughNet (MS)	HG-104	ExtremeNet	512×512	≤1.8×	46.4	**65.1**	**50.7**	**29.1**	48.5	58.1	–

4.4   Comparison with the state-of-the-art

For comparison with the state-of-the-art, we use Hourglass-104 [23] backbone. We train Hourglass model with a batch size of 36 for 100 epochs using the Adam optimizer [21]. We set the initial learning rate to 2.5×10^{-4} and divided it by 10 at epoch 90. Table 5 presents performances of HoughNet and several established state-of-the-art detectors. First, we compare HoughNet with OAP [51] since it is the model on which we built HoughNet. In OAP, they did not present any results for "from scratch" training. Instead they fine-tuned their model from ExtremeNet weights. When we do the same (i.e. initialize HoughNet with ExtremeNet weights), we obtain better results than OAP. However as expected, HoughNet is slower than OAP. Among the one-stage bottom-up object detectors, HoughNet performs on-par with the best bottom-up object detector by achieving 46.4 AP against 47.0 AP of CenterNet [9]. HoughNet outperforms CenterNet on AP_{50} (65.1 AP_{50} vs. 64.5 AP_{50}). Note that, since our model is initialized with ExtremeNet weights, which makes use of the segmentation masks in its own training, our model effectively uses more data compared to CenterNet. HoughNet is the fastest among one-stage bottom-up detectors. It is faster than CenterNet, CornerNet and more than twice as fast as ExtremeNet.

We provide visualization of votes for sample detections of HoughNet for qualitative visual inspection (Fig. 4). These detections clearly show that HoughNet is able to make use of long-range visual evidence.

4.5   Using our voting module in another task

One task where long-range interactions could be useful is the task of image generation from a given label map. There are two main approaches to solve this task; using unpaired and paired data for training. We take CycleGAN [54] and Pix2Pix [20] as our baselines for unpaired and paired approaches, respectively. We attach our voting module at the end of CycleGAN [54] and Pix2Pix [20] models.

Table 6: Comparison of FCN scores for the "labels to photo" task on the Cityscapes [7] dataset

Method	Per-pixel acc.	Per-class acc.	Class IOU
CycleGAN	0.43	0.14	0.09
+ Voting	**0.52**	**0.17**	**0.13**
pix2pix	0.71	**0.25**	0.18
+ Voting	**0.76**	**0.25**	**0.20**

For quantitative comparison, we use the Cityscapes [7] dataset. In Table 6, we present FCN scores [31] (which is used as the measure of success in this task) of CycleGAN and Pix2Pix with and without our voting module. To obtain the "without" result, we used the already trained model shared by the authors. We obtained the "with" result using the official training code from their repositories. In both cases evaluation was done using the official test and evaluation scripts from their repos. Results show that using the voting module improves FCN scores by large margins. Qualitative inspection also shows that when our voting module is attached, the generated images conform to the given input segmentation maps better (Fig. 5). This is the main reason for the quantitative improvement. Since Pix2Pix is trained with paired data, generated images follow input segmentation maps, however, Pix2Pix fails to generate small details.

Detection	Voters	Detection	Voters	Detection	Voters

Fig. 4: Sample detections of HoughNet and their vote maps. In the "detection" columns, we show a correctly detected object, marked with a yellow bounding box. In the "voters" columns, the locations that vote for the detection are shown. Colors indicate vote strength based on the standard "jet" colormap (red is high, blue is low; Fig. 1). In the **top row**, there are three "mouse" detections. In all cases, in addition to the local votes (that are on the mouse itself), there are strong votes coming from nearby "keyboard" objects. This voting pattern is justified given that mouse and keyboard objects frequently co-appear. A similar behavior is observed in the detections of "baseball bat", "baseball glove" and "tennis racket" in the **second row**, where they get strong votes from "ball" objects that are far-away. Similarly, in the **third row**, "vase" detections get strong votes from the flowers. In the first example of the **bottom row**, "dining table" detection gets strong votes from the candle object, probably because they co-occur frequently. Candle is not among the 80 classes of COCO dataset. Similarly, in the second example in the **bottom row**, "dining table" has strong votes from objects and parts of a standard living room. In the last example, partially occluded bird gets strong votes (stronger than the local votes on the bird itself) from the tree branch

| Input | CycleGAN | + Voting | Input | Pix2Pix | + Voting |
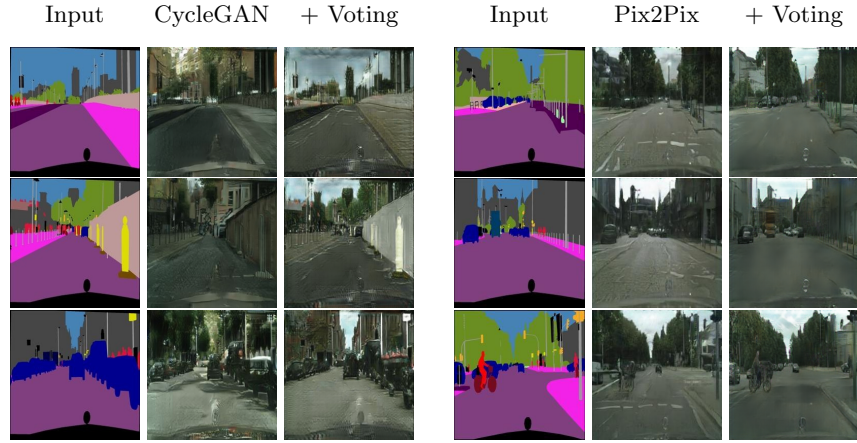
Fig. 5: Sample qualitative results for the "labels to photo" task. When integrated with CycleGAN, our voting module helps generate better images in the sense that the image conforms to the input label map better. In all three images, CycleGAN fails to generate sky, buildings and falsely generates vegetation in the last image. When used with Pix2Pix, it helps generate more detailed images. In the first row, cars and buildings can be barely seen for Pix2Pix. Similarly, a bus is generated as a car and a bicycle is silhouetted in the second and third images, respectively. Our voting module fixes these errors

5   Conclusion

In this paper, we presented HoughNet, a new, one-stage, anchor-free, voting-based, bottom-up object detection method. HoughNet determines the presence of an object at a specific location by the sum of the votes cast on that location. Voting module of HoughNet is able to use both short and long-range evidence through its log-polar vote field. Thanks to this ability, HoughNet generalizes and enhances current object detection methodology, which typically relies on only local (short-range) evidence. We show that HoughNet performs on-par with the state-of-the-art bottom-up object detector, and obtains comparable results with one-stage and two-stage methods. To further validate our proposal, we used the voting module of HoughNet in an image generation task. Specifically, we showed that our voting module significantly improves the performance of two GAN models in a "labels to photo" task.

References

1. Akbas, E., Eckstein, M.P.: Object detection through search with a foveated visual system. PLoS computational biology **13**(10), e1005743 (2017)
2. Ballard, D.H., et al.: Generalizing the hough transform to detect arbitrary shapes. Pattern Recognition (1981)
3. Barinova, O., Lempitsky, V., Kholi, P.: On detection of multiple object instances using hough transforms. IEEE Transactions on Pattern Analysis and Machine Intelligence **34**(9), 1773–1784 (2012)
4. Belongie, S., Malik, J., Puzicha, J.: Shape matching and object recognition using shape contexts. IEEE Transactions on Pattern Analysis and Machine Intelligence **24**(4), 509–522 (April 2002). https://doi.org/10.1109/34.993558
5. Bodla, N., Singh, B., Chellappa, R., Davis, L.S.: Soft-nms–improving object detection with one line of code. In: IEEE International Conference on Computer Vision. pp. 5561–5569 (2017)
6. Cai, Z., Vasconcelos, N.: Cascade R-CNN: Delving into high quality object detection. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 6154–6162 (2018)
7. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 3213–3223 (2016)
8. Dai, J., Li, Y., He, K., Sun, J.: R-FCN: Object detection via region-based fully convolutional networks. In: Advances in Neural Information Processing Systems. pp. 379–387 (2016)
9. Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., Tian, Q.: Centernet: Keypoint triplets for object detection. In: IEEE International Conference on Computer Vision (2019)
10. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part based models. IEEE Transactions on Pattern Analysis and Machine Intelligence (2010)
11. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. IEEE Transactions on Pattern Analysis and Machine Intelligence **32**(9), 1627–1645 (2009)
12. Fu, C.Y., Liu, W., Ranga, A., Tyagi, A., Berg, A.C.: Dssd: Deconvolutional single shot detector. arXiv preprint arXiv:1701.06659 (2017)
13. Gabriel, E., Schleiss, M., Schramm, H., Meyer, C.: Analysis of the discriminative generalized hough transform as a proposal generator for a deep network in automatic pedestrian and car detection. Journal of Electronic Imaging **27**(5), 051228 (2018)
14. Gall, J., Lempitsky, V.: Class-specific hough forests for object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (2009)
15. Girshick, R.B., Felzenszwalb, P.F., McAllester, D.: Discriminatively trained deformable part models, release 5. http://people.cs.uchicago.edu/ rbg/latent-release5/
16. He, K., Gkioxari, G., Dollár, P., Girshick, R.B.: Mask r-cnn. IEEE International Conference on Computer Vision pp. 2980–2988 (2017)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)

18. Hough, P.V.C.: Machine Analysis of Bubble Chamber Pictures **C590914**, 554–558 (1959)
19. Hu, H., Gu, J., Zhang, Z., Dai, J., Wei, Y.: Relation networks for object detection. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 3588–3597 (2018)
20. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1125–1134 (2017)
21. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
22. Land, M., Tatler, B.: Looking and acting: vision and eye movements in natural behaviour. Oxford University Press (2009)
23. Law, H., Deng, J.: Cornernet: Detecting objects as paired keypoints. In: European Conference on Computer Vision. pp. 734–750 (2018)
24. Leibe, B., Leonardis, A., Schiele, B.: Robust object detection with interleaved categorization and segmentation. International Journal of Computer Vision **77**(1), 259–289 (May 2008)
25. Lifshitz, I., Fetaya, E., Ullman, S.: Human pose estimation using deep consensus voting. In: European Conference on Computer Vision (2016)
26. Lin, T., Dollár, P., Girshick, R.B., He, K., Hariharan, B., Belongie, S.J.: Feature pyramid networks for object detection. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 936–944 (2017)
27. Lin, T., Goyal, P., Girshick, R.B., He, K., Dollár, P.: Focal loss for dense object detection. In: IEEE International Conference on Computer Vision (2017)
28. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: European Conference on Computer Vision. pp. 740–755. Springer (2014)
29. Liu, S., Qi, L., Qin, H., Shi, J., Jia, J.: Path aggregation network for instance segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 8759–8768 (2018)
30. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European Conference on Computer Vision (2016)
31. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 3431–3440 (2015)
32. Maji, S., Malik, J.: Object detection using a max-margin hough transform. In: IEEE Conference on Computer Vision and Pattern Recognition (2009)
33. Okada, R.: Discriminative generalized hough transform for object detection. In: IEEE International Conference on Computer Vision (2009)
34. Oksuz, K., Cam, B., Akbas, E., Kalkan, S.: Localization recall precision (lrp): A new performance metric for object detection. In: European Conference on Computer Vision (ECCV) (2018)
35. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems, pp. 8024–8035. Curran Associates, Inc. (2019), http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

36. Qi, C.R., Litany, O., He, K., Guibas, L.J.: Deep hough voting for 3d object detection in point clouds. In: IEEE International Conference on Computer Vision (2019)
37. Razavi, N., Gall, J., Kohli, P., Van Gool, L.: Latent hough transform for object detection. In: European Conference on Computer Vision (2012)
38. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. In: IEEE Conference on Computer Vision and Pattern Recognition (2017)
39. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767 (2018)
40. Ren, J., Chen, X., Liu, J., Sun, W., Pang, J., Yan, Q., Tai, Y.W., Xu, L.: Accurate single stage detector using recurrent rolling convolution. In: IEEE Conference on Computer Vision and Pattern Recognition (2017)
41. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems. pp. 91–99 (2015)
42. Sheshkus, A., Ingacheva, A., Arlazarov, V., Nikolaev, D.: Houghnet: neural network architecture for vanishing points detection (2019)
43. Tian, Z., Shen, C., Chen, H., He, T.: Fcos: Fully convolutional one-stage object detection. In: IEEE International Conference on Computer Vision (2019)
44. Traver, V.J., Bernardino, A.: A review of log-polar imaging for visual perception in robotics. Robotics and Autonomous Systems **58**(4), 378–398 (2010)
45. Wang, J., Chen, K., Yang, S., Loy, C.C., Lin, D.: Region proposal by guided anchoring. In: IEEE Conference on Computer Vision and Pattern Recognition (2019)
46. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 7794–7803 (2018)
47. Xiao, B., Wu, H., Wei, Y.: Simple baselines for human pose estimation and tracking. In: European Conference on Computer Vision. pp. 466–481 (2018)
48. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. CoRR (2015)
49. Zhang, S., Wen, L., Bian, X., Lei, Z., Li, S.Z.: Single-shot refinement neural network for object detection. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 4203–4212 (2018)
50. Zhang, X., Wan, F., Liu, C., Ji, R., Ye, Q.: Freeanchor: Learning to match anchors for visual object detection. In: Advances in Neural Information Processing Systems (2019)
51. Zhou, X., Wang, D., Krähenbühl, P.: Objects as points. In: arXiv preprint arXiv:1904.07850 (2019)
52. Zhou, X., Zhuo, J., Krähenbühl, P.: Bottom-up object detection by grouping extreme and center points. In: IEEE Conference on Computer Vision and Pattern Recognition (2019)
53. Zhu, C., He, Y., Savvides, M.: Feature selective anchor-free module for single-shot object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (2019)
54. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: IEEE International Conference on Computer Vision. pp. 2223–2232 (2017)
55. Zhu, Y., Zhao, C., Wang, J., Zhao, X., Wu, Y., Lu, H.: Couplenet: Coupling global structure with local parts for object detection. In: IEEE International Conference on Computer Vision. pp. 4126–4134 (2017)