# Gradient Corner Pooling for Keypoint-Based Object Detection

**Xuyang Li[1], Xuemei Xie[1, 2*], Mingxuan Yu[1], Jiakai Luo[1], Chengwei Rao[1], Guangming Shi[1, 3]**

[1] Xidian University, Xi'an, 710071, China.
[2] Pazhou Lab, Huangpu, 510555, China.
[3] Peng Cheng Laboratory, Shenzhen, 518055, China.
{xylee, mxuanyu, jkluo, cwrao}@stu.xidian.edu.cn, xmxie@mail.xidian.edu.cn, gmshi@xidian.edu.cn

## Abstract

Detecting objects as multiple keypoints is an important approach in the anchor-free object detection methods while corner pooling is an effective feature encoding method for corner positioning. The corners of the bounding box are located by summing the feature maps which are max-pooled in the x and y directions respectively by corner pooling. In the unidirectional max pooling operation, the features of the densely arranged objects of the same class are prone to occlusion. To this end, we propose a method named Gradient Corner Pooling. The spatial distance information of objects on the feature map is encoded during the unidirectional pooling process, which effectively alleviates the occlusion of the homogeneous object features. Further, the computational complexity of gradient corner pooling is the same as traditional corner pooling and hence it can be implemented efficiently. Gradient corner pooling obtains consistent improvements for various keypoint-based methods by directly replacing corner pooling. We verify the gradient corner pooling algorithm on the dataset and in real scenarios, respectively. The networks with gradient corner pooling located the corner points earlier in the training process and achieve an average accuracy improvement of 0.2%-1.6% on the MS-COCO dataset. The detectors with gradient corner pooling show better angle adaptability for arrayed objects in the actual scene test.

## Introduction

Object detection is the foundation of a large number of vision tasks, such as automatic driving, scene understanding, and video surveillance. According to the method that the bounding box proposed, there are currently two mainstream frameworks for object detection. One is the anchor-based method. These methods generate a large number of candidate anchors according to certain image features or rules and have the highest detection accuracy on object detection datasets such as MS-COCO (Lin et al. 2014). The other is the anchor-free method, represented by a series of papers such as FCOS (Tian et al. 2019), CornerNet (Law and Deng 2018; Law et al. 2019), ExtremeNet (Zhou, Zhuo, and Krahenbuhl 2019), Objects as Points (Zhou, Wang, and Krahenbuhl 2019), RepPoints (Yang et al. 2019; Chen et al. 2020), CenterNet (Duan et al. 2019) and etc. Among these methods,

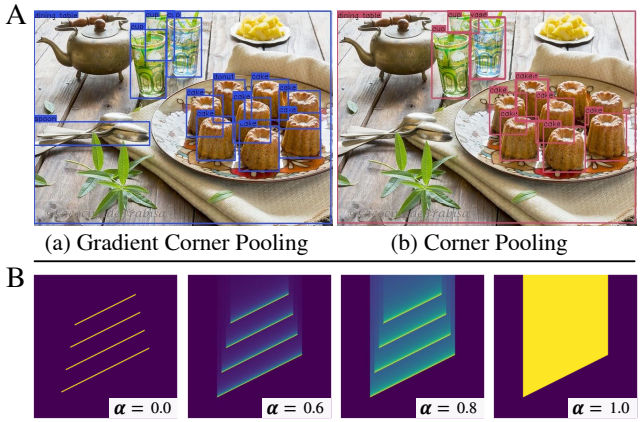



Figure 1: A shows the detection result of an image in the MS-COCO validation dataset. (a) is the detection result of CornerNet-Saccade with gradient corner pooling (GCP). (b) is the result of this detector with corner pooling. There is a significant improvement in the performance of detecting array-placed objects for keypoint-based detectors with GCP. B is the illustration of top direction in the gradient corner pooling, which encodes the distance in directional max pooling with the modulation factor $\alpha$.

CornerNet (Law and Deng 2018) proposes a novel model, which detects an object as a set of paired corner points. The width and height of the bounding box of the object are directly calculated by the paired top-left and bottom-right corner points.

Corner pooling consists of max pooling in four directions: top pooling, left pooling, bottom pooling, and right pooling. The pooling results in the two directions are added respectively to obtain top-left pooling and bottom-right pooling. This encoding method diverts the maximum points of the object feature response on the feature map to the top-left and bottom-right corners. However, max pooling in one direction leads to the occlusion in feature responses of objects of the same class in this direction, which in turn makes corner detection difficult. The essential reason is that max pooling loses part of the spatial information on feature maps. Based on the corner pooling, we propose gradient corner pooling

*Corresponding author.

Algorithm 1: Gradient corner pooling

**Require:** $F_{ij}$

$F_{ij}$ is a batch of feature maps with dimension Height $\times$ Width
$X, Y, R \leftarrow F_{ij}$
$n = 0$
**while** $n < \log_2 H$ **do**
    $Y^H_{0:H-2^n} \leftarrow max(X^H_{0:H-2^n}, X^H_{2^n:H})$
    $X \leftarrow Y$ ▷ Corner pooling
    $R \leftarrow \alpha' R + Y$ ▷ Gradient corner pooling
**end while**
**return** $R$

(GCP), which includes a modulation factor to encode the spatial distance information of features during the pooling process. The point with the largest value will be multiplied by the coefficient of distance and be moved to the current point in the process of four directions max pooling. Such a proportional decrease enables other similar objects features within the coverage of the direction pooling in the corner pooling to be highlighted, which greatly alleviates the feature occlusion problem of homogeneous object features nearby. The top pooling in gradient corner pooling of different modulation factors is shown in Figure 1B. This encoding algorithm can be computed efficiently by dynamic programming with only adding one line of code, which is shown in Algorithm 1. $X, Y$ and $R$ are initialized with a batch of feature maps $F_{ij}$, and they are exactly the same. The subscript $i, j$ represent the index of the height and width of the feature maps. The arrow represents the directional assignment.

In order to alleviate the influence of the distance decay brought by the modulation factor, we design a set of handcrafted convolution kernels to discover fixed patterns of corners, providing better detection performance for densely arranged objects with various angles. To demonstrate the effectiveness of the proposed gradient corner pooling, we evaluate the performance of the method on the MS-COCO dataset and in real scenes in Section 4.

CornerNet-Saccade is regarded as a baseline in this article, which is more efficient in training and inference than CornerNet. GCP is also used for optimizing the attention module in CornerNet-Saccade. GCP encodes the center point in attention mechanism to the corner point, and we named it corner attention. The corner attention and prediction head with GCP act on the feature map generated by backbone in a consistent manner, which can significantly improve the recall rate by 2.1%.

The gradient corner pooling, as an alternative to the original corner pooling algorithm in the current keypoint-based object detection pipeline, brings about 0.8%, 1.1%, 0.5%, 0.3% average accuracy improvements on CornerNet, CornerNet-Saccade, CenterNet, and RepPoints v2, respectively. The networks show better angle adaptability on arrayed objects with GCP in the actual scene test.

Our contributions are three-fold:

- We propose gradient corner pooling, an encoding method with the spatial distance of homogeneous object features embedded for corner localization. Our method is concise and general, it can be applied to keypoint-based detectors which detect corners without increasing inference time.
- We propose a fixed pattern of finding corners by combining gradient corner pooling and handcrafted convolution kernels, which is well interpretable.
- Corner attention is proposed to keep the optimization of attention mechanism and prediction head consistent for the backbone. In this way, performance of the saccade mechanism is improved.

The remainder of the article is organized as follows: in the next section, we discuss the related object detectors using corner pooling. In Sec. 3 we describe the expression and calculation process of gradient corner pooling. Then we analyze the roles of the handcrafted convolution kernel and corner attention. Evaluations of our methods on datasets and in real scenarios are presented in Sec. 4. Finally, in Sec. 5, we summarize our work and give conclusions.

## Related Work

According to the different forms of proposals, object detection approaches powered by deep neural networks can be divided into two main types: anchor-based methods and anchor-free methods. The size and shape of the rectangular representation of the object are usually pre-defined for anchor-based methods. These anchor boxes distributed on the feature map learn scale information from the object features. The representative methods are Fast R-CNN (Girshick 2015), Faster R-CNN (Ren et al. 2015), SSD (Liu et al. 2016), YOLOv2 (Redmon and Farhadi 2017), Mask R-CNN (He et al. 2017), RetinaNet (Lin et al. 2017), TridentNet (Li et al. 2019), Grid R-CNN (Lu et al. 2019), RefineDet (Zhang et al. 2018), Cascade R-CNN (Cai and Vasconcelos 2018), and the others in YOLO series (Bochkovskiy, Wang, and Liao 2020; Ge et al. 2021; Wang, Bochkovskiy, and Liao 2022). Scaled-YOLOv4 (Wang, Bochkovskiy, and Liao 2021) and EfficientDet series (Tan, Pang, and Le 2020) propose a compound scaling method that uniformly scales the resolution, depth, and width for all backbone, feature network, bounding box and class prediction networks at the same time. A large number of anchor boxes can cover adjacent or even partial occluded objects, which is one of the reasons that anchor-based methods perform well on object detection tasks. YOLO (Redmon et al. 2016) and FCOS (Tian et al. 2019) are the earlier anchor-free methods, YOLO (Redmon et al. 2016) predicts bounding box coordinates directly from an image and is later improved in YOLOv2 (Redmon and Farhadi 2017) by switching to anchor boxes. FCOS (Tian et al. 2019) detects all the positive points and learns the distance from the point to the border of the bounding box to predict height and width.

**Detecting object as keypoints.** CornerNet (Law and Deng 2018) proposes the approach to detect an object bounding box as a pair of keypoints, which makes the anchor-free method directly get the bounding box from the paired keypoints. Subsequently, Objects as Points (Zhou, Wang, and Krahenbuhl 2019) uses keypoint estimation to

find center points and regresses to all other object properties, such as size, 3D location, orientation. CenterNet (Duan et al. 2019) designs cascade corner pooling and center pooling to detect each object as a triplet of keypoints: two corner points and a center point. Extreme Clicking (Papadopoulos et al. 2017) labels the object with extreme points in four directions instead of width and height of the bounding box. ExtremeNet (Zhou, Zhuo, and Krahenbuhl 2019) detects four extreme points (top-most, left-most, bottom-most, right-most) and one center point of the object using a standard keypoint estimation network, and groups the five keypoints into a bounding box if they are geometrically aligned. RepPoints (Yang et al. 2019) proposes a new finer representation of objects as a set of sample points useful for both localization and recognition.

**Keypoints prediction with corner pooling.** CornerNet proposes a novel corner pooling, which optimizes the positioning of keypoints. This method provides more candidate boxes with high IOU and improves the network performance by about 2.0% of average precision than the max pooling on the MS-COCO dataset. Corner pooling aims to find the maximum values on the boundary and encode them into the corners. This method is widely used in keypoint-based state-of-the-art object detection methods such as CornerNet-Lite (Law et al. 2019), CPNDet (Duan et al. 2020), RepPoints v2 (Chen et al. 2020), CenterNet (Duan et al. 2019), CentripetalNet (Dong et al. 2020) and CenterNet++ (Duan et al. 2022). CornerNet-Saccade designs an attention mechanism to eliminate the need for exhaustively processing all pixels of the image, and improves the efficiency of CornerNet by 6.0× and the AP by 1.0% on MS-COCO. By combining the corner pooling in different directions, CenterNet and CenterNet++ propose cascade corner pooling and center pooling. Cascade corner pooling enables corners to extract features from central regions of the object. As a basic encoding module that assists corner points locating, corner pooling is very important in the solution of object detection as multiple keypoints.

## Approach

Spatial distance information is prior knowledge of feature maps. Gradient corner pooling is an encoding method that adds such prior knowledge in directional max pooling. We employ gradient corner pooling to solve the possible interference problem of arrayed similar objects on the feature map in the process of encoding information. Object features of the same class nearby will lead to occlusion when looking horizontally towards the right for the topmost boundary of an object and vertically towards the bottom for the leftmost boundary on the feature map. Objects of the same class can be clearly distinguished on the feature map by encoding features and spatial distances. We propose the modulation factor $\alpha$ of the distance between the current point and the pooling point to adjust the value of the corner point pooling process according to the distance. This spatial distance encoding process requires only one line of code added to the original corner pooling in the actual implementation process. Then, we perform corner pattern discovery via fixed handcrafted kernels which significantly enhance the model's adaptability to angles. In the end, corner attention is adopted to ensure the consistency of features extracted by the backbone and prediction head, which improves the recall and accuracy of the model.

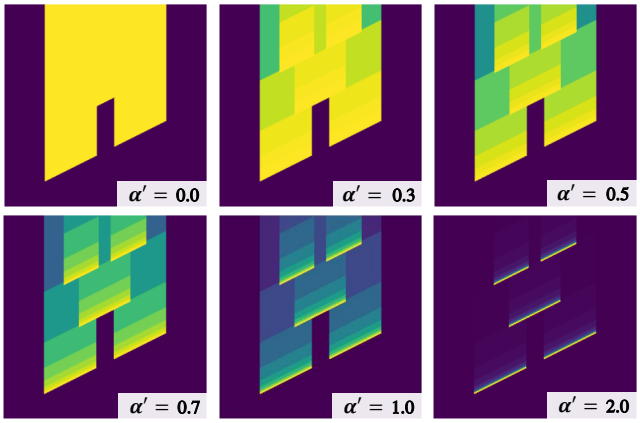



Figure 2: Illustration of the gradient top pooling. This figure shows the simulated feature maps of the object array at different modulation coefficients, which calculated by algorithm 1. The objects feature after pooling is the same as the original feature when the modulation factor $\alpha$ is zero. As the modulation factor increases, the value at the point away from the unidirectional maximum value increases.

### Gradient Corner Pooling

A detected object is represented by a bounding box in object detection. The bounding box is usually represented by two points on the top-left and bottom-right. Corner pooling (Law and Deng 2018) can re-encode the feature map to obtain accurate corner positions. Gradient corner pooling is designed to address the occlusion problem of homogeneous object features in the encoding process of the directional max pooling. The computation of top and left pooling are expressed by the following equations:

$$t_{ij} = \begin{cases} max(f_{t_{ij}}, t_{(i+1)j}) & \text{if } i < H \\ f_{t_{Hj}} & \text{otherwise} \end{cases} \quad (1)$$

$$l_{ij} = \begin{cases} max(f_{l_{ij}}, l_{i(j+1)}) & \text{if } j < W \\ f_{l_{iW}} & \text{otherwise} \end{cases} \quad (2)$$

In the above $f_{t_{ij}}$ and $f_{l_{ij}}$ are defined as the vectors at location $(i, j)$ in feature map $f_t$ and $f_l$ which are the input of top-left corner pooling. $t_{ij}$ and $l_{ij}$ are the outputs. $H$ and $W$ represent the height and width of the feature maps. The corner pooling layer max-pools all feature vectors between $(i, j)$ and $(i, H)$ in $f_t$ to a feature vector $t_{ij}$, and max-pools all feature vectors between $(i, j)$ and $(W, j)$ in $f_l$ to a feature vector $l_{ij}$.

As Figure 2 ($\alpha' = 0$) shows, the boundary of objects feature disappeared during the max pooling in the top direction, making it hard to differentiate objects of the same class above. The modulating factor $\alpha^{\log_2 d}$ is related to the spatial distance in each direction of the corner pooling. As shown

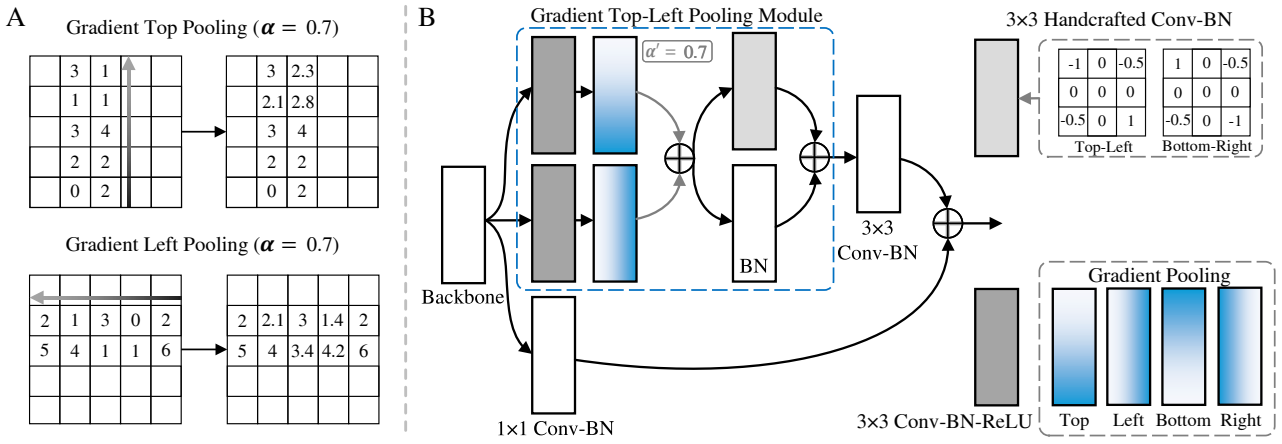



Figure 3: A shows calculated value of the gradient top-left pooling. B shows the pipeline of gradient top-left pooling module. We replace the original corner pooling with gradient corner pooling and add a group of handcrafted convolution kernels after the corner pooling to identify fixed corner patterns.

in Eq. 3 and Eq. 4, $d_{f_{t_{ij}}}, d_{f_{l_{ij}}}$ represent the pixel distance on the feature map from location $(i, j)$ to the location of the row or column with the maximum feature value after modulation. We define the gradient top-left corner pooling as:

$$f(d_{f_{l_{ij}}}) = \alpha^{\log_2(d_{f_{l_{ij}}}+1)} \tag{3}$$

$$f(d_{f_{t_{ij}}}) = \alpha^{\log_2(d_{f_{t_{ij}}}+1)} \tag{4}$$

$$t_{ij} = \begin{cases} max(f_{t_{ij}}, f(d_{f_{t_{ij}}}) t_{(i+d_{f_{t_{ij}}})j}) & \text{if } i < H \\ f_{t_{Hj}} & \text{otherwise} \end{cases} \tag{5}$$

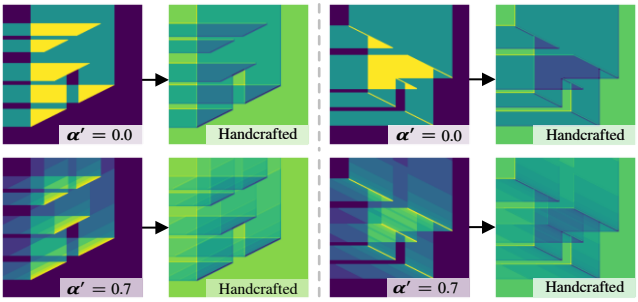
$$l_{ij} = \begin{cases} max(f_{l_{ij}}, f(d_{f_{l_{ij}}}) l_{i(j+d_{f_{l_{ij}}})}) & \text{if } j < W \\ f_{l_{iW}} & \text{otherwise} \end{cases} \tag{6}$$

When the modulation factor $\alpha = 1$, the computation above is exactly the same as corner pooling. As $\alpha$ decreases, the feature maps that pass through the gradient corner pooling layer are more similar to the input features. GCP is visualized for several values of $\alpha \in (0, 1]$ in Figure 1B. The gradient bottom-right pooling is defined in a similar way.

As shown in algorithm 1, in the code implementation, the parallel comparison calculation is performed in blocks, so the actual distance modulation changes with the distance of an exponential of 2. The relationship between the fast calculate modulation factor $\alpha'$ and $\alpha$ is $\alpha' = (1-\alpha)/\alpha, \alpha \in (0, 1]$ and $\alpha' \in [0, +\infty)$. The fast computation result of top direction in gradient corner pooling is shown in Figure 2. Gradient corner pooling effectively improves the detection performance of object detection algorithms on arrayed objects of the same class.

## Object Angle and Handcrafted Convolution Kernel

Due to the presence of modulation, GCP and CP present different values at the corners. Objects with different angles




(a) Right oblique object (b) Left oblique object

Figure 4: (a) is corner recognition of handcrafted convolution kernel for right oblique objects on the simulated feature map while (b) is for left oblique objects. GCP and handcrafted convolution kernels improve the performance of locating the corners of objects in different oblique directions.

on the feature map exhibit different patterns during gradient corner pooling. In the method of objects as paired keypoints, when the upper part of the object is close to the top-left corner and the lower part is close to the bottom-right corner, which is regarded as the left oblique object, the corner points are easier to be detected. On the contrary, the top-left and bottom-right corners are not the positions where the feature map is lighted up significantly after gradient corner pooling when the object is right oblique. We note two approximately fixed patterns of the top-left and bottom-right corners on the feature map and design two handcrafted convolution kernels to extract features of such patterns. As shown in Figure 4, the handcrafted convolution kernels extract the keypoints at the top-left and bottom-right corners, respectively. Gradient corner pooling and handcrafted convolution kernels extract the localization information of the corners while the values on the entire feature map change non-linearly, which leads to the loss of partial semantic information. To this end, the

feature map branch that pass through GCP is added for effective information completion. The final GCP module with handcrafted convolution kernels is shown in Figure 3B.

### Corner Attention with Gradient Corner Pooling

CornerNet-Saccade detects objects within small regions around possible object locations in an image. It uses the downsized full image to predict attention maps and coarse bounding boxes which both suggest possible object locations. CornerNet-Saccade then detects objects by examining the regions centered at the locations in high resolution. Saccade is an effective mechanism for small object detection. However, the final prediction head pushes the backbone network to learn more features about the edges while the attention mechanism makes the network pay more attention to the center point of the object. There needs to be a balance between center point attention and prediction head for corner points. We add the GCP module to the attention branch to encode the attention map, which ensures the corner attention and prediction head act on the backbone in the same way. An overview of the corner attention is shown in Figure 5. For a downsized image, corner attention predicts 6 attention maps, two for small objects, two for medium objects and two for large objects. Each two feature maps are the top-left corner and bottom-right corner of the object, respectively. The feature maps are obtained from the backbone network in CornerNet-Saccade, which is an hourglass network (Newell, Yang, and Deng 2016). The feature maps from the upsampling layers in the hourglass are used to predict the attention maps. The feature maps at finer scales are used for smaller objects and the ones at coarser scales are for larger objects. We predict the corner attention maps by applying a $3 \times 3$ Conv-ReLU module followed by GCP modules and a $1 \times 1$ Conv-Sigmoid module to each feature map. During training, we set the top-left and bottom-right corner of each bounding box on the corresponding attention map positive and the rest negative. During the inference, we crop the image at the lower right position of the point in the top-left attention map and crop at upper left based on the bottom-right attention map. NMS is adopted to remove redundant locations. We apply the focal loss, which is the same as CornerNet-Saccade.

## Experiments

We test our method on the dataset and real-world scenarios respectively and conduct ablation experiments for each module.

### Dataset, Metrics and Baseline

We evaluate our method on the MS-COCO dataset (Lin et al. 2014). We follow common practice (Lin et al. 2017) and use the COCO trainval35k split (union of 80k images from train and a random 35k subset of images from the 40k image val split) for training networks such as CornerNet, CornerNet-Saccade, CenterNet and CenterNet++. The testing of networks is performed on the test-dev 2017, which has no public labels and requires the use of the evaluation server.

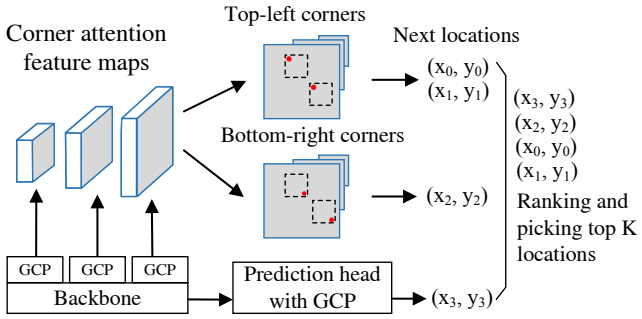



Figure 5: Illustration of corner attention module. Each scale's corner attention maps consist of the top-left attention map and the bottom-right attention map. In the inference process, the center $(x, y)$ of the next location needs to be caculated according to the scale and top-left point or bottom-right point, respectively. Then non-maximum suppresion (NMS) operations are performed together to remove redundancy.

The ablation studies and visualization experiments are performed on the corresponding validation set. To evaluate the effectiveness of GCP module, we construct a subset of arrayed objects to quantitatively compare the performance of GCP module and original corner pooling. We select the images with arrayed objects in MS-COCO val2017 to form a subset of 1251 images. There are more than five objects of the same class in each of these images. We also construct an arrayed objects subset consisting of images in real sence. These images are rotated in $5^\circ$ intervals to verify the angular adaptability of GCP module. There are a total of 3528 images in this subset. Part of them are shown in Figure 7.

We also test GCP module on the arrayed objects subset of the MS-COCO val2017. GCP module obtains more improvements on the arrayed objects subset.

### Comparison with Keypoint-Based Models

As shown in Table 1, to evaluate the proposed method, we test GCP with four state-of-the-art keypoint-based detectors, namely CornerNet-Saccade, CenterNet, CenterNet++ and RepPoints v2. We replace the corner pooling with the GCP module in the network and retrain these networks with their original initialization. The proposed gradient corner pooling module achieves improvements. Our method achieves 0.8% AP improvement compared to the CornerNet (Hourglass-104) with corner pooling. Improvements of 0.3% and 1.1% AP are achieved on RepPoints v2 and CornerNet-Saccade with GCP module. We also report the detection results of CenterNet and CenterNet++ using gradient corner pooling, which obtain improvements of 0.5% and 0.2% with the backbone of Hourglass-52 and ResNet-50, respectively. A comparison of some detections is shown in Figure 6. For all the experiments, due to limited resources, we train CornerNet-104, CenterNet-52, CenterNet++, RepPoints v2, CornerNet-Saccade on four GeForce RTX3090 GPUs and follow the training details in (Law and Deng 2018; Duan et al. 2019, 2022; Chen et al. 2020; Law et al. 2019), respec-

| Method | Backbone | $AP$ | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|
| **keypoint-based:** | | | | | | | |
| Objects as Points (ss) | Hourglass-104 | 42.1 | 61.1 | 45.9 | 24.1 | 45.5 | 52.8 |
| FCOS (Tian et al. 2019) | ResNet-101-FPN | 41.5 | 60.7 | 45.0 | 24.4 | 44.8 | 51.6 |
| RepPoints (ss) (Yang et al. 2019) | ResNet-101 | 41.0 | 62.9 | 44.3 | 23.6 | 44.1 | 51.7 |
| RepPoints v2 (ss) (Chen et al. 2020) | ResNet-50 (24 epoch) | 44.4 | 63.5 | 47.7 | 26.6 | 47.0 | 54.6 |
| CPNDet (ss) (Duan et al. 2020) | DLA-34 | 41.7 | 44.9 | 20.2 | 23.9 | 44.1 | 56.4 |
| CPNDet (ss) (Duan et al. 2020) | Hourglass-104 | 47.0 | 65.0 | 51.0 | 26.5 | 50.2 | 60.7 |
| CentripetalNet (ss) (Dong et al. 2020) | Hourglass-104 | 45.8 | 63.0 | 49.3 | 25.0 | 48.2 | 58.7 |
| CornerNet511 (ss) (Law and Deng 2018) | Hourglass-52 | 37.8 | 53.7 | 40.1 | 17.0 | 39.0 | 50.5 |
| CornerNet511 (ss) (Law and Deng 2018) | Hourglass-104 | 40.6 | 56.4 | 43.2 | 19.1 | 42.8 | 54.3 |
| CornerNet511 (ms) (Law and Deng 2018) | Hourglass-104 | 42.2 | 57.8 | 45.2 | 20.7 | 44.8 | 56.6 |
| CornerNet-Saccade (Law et al. 2019) | Hourglass-54 | 43.2 | 59.1 | 46.5 | 25.0 | 44.7 | 56.7 |
| CenterNet511 (ss) (Duan et al. 2019) | Hourglass-52 | 41.6 | 59.4 | 44.2 | 22.5 | 43.1 | 54.1 |
| CenterNet++ (ss) (Duan et al. 2022) | ResNet-50 (24 epoch) | 46.4 | 63.7 | 50.3 | 27.1 | 48.9 | 58.8 |
| CornerNet511 (ss) w/ GCP† | Hourglass-104 | 41.4 (+0.8) | 57.3 | 43.8 | 20.9 | 44.1 | 55.6 |
| CornerNet511 (ms) w/ GCP† | Hourglass-104 | 42.8 (+0.6) | 58.5 | 46.0 | 22.3 | 45.3 | 57.1 |
| CornerNet-Saccade w/ GCP‡ | Hourglass-54 | 44.3 (+1.1) | 60.8 | 47.9 | 26.5 | 45.6 | 57.6 |
| CenterNet511 w/ GCP† | Hourglass-52 | 42.1 (+0.5) | 59.6 | 44.8 | 23.4 | 43.6 | 54.5 |
| RepPoints v2 (ss) w/ GCP† | ResNet-50 (24 epoch) | 44.7 (+0.3) | 64.1 | 48.0 | 27.2 | 47.3 | 54.6 |
| CenterNet++ (ss) w/ GCP† | ResNet-50 (24 epoch) | 46.6 (+0.2) | 64.0 | 50.4 | 27.3 | 49.0 | 59.1 |

Table 1: Performance(%) comparison of keypoint-based methods on the MS-COCO test-dev dataset. The abbreviations are: 'ss' − singe-scale testing, 'ms' − multi-scale testing, 'w/ GCP' – training and inference with gradient corner pooling, † − add handcrafted convolution kernel, ‡ − replace center attention with corner attention.

tively.

We rotate the images shot in the actual scene to observe the sensitivity of the detector on various object angles with the corner pooling and GCP module. As shown in Figure 7, the results suggest that the encoding ways in GCP significantly improve the performance of detecting arrayed objects in the pooling process, and enhance the adaptability of the detector for various angles.

## Ablation Study

The main contribution of the GCP module is to solve the occlusion problem of homogeneous object features through distance encoding, which improves the performance of arrayed objects detection. We further evaluate the GCP module on the arrayed objects subset of the MS-COCO val2017. As shown in Table 2, as the number of arrayed objects increases, the performance of CornerNet-Saccade with the GCP module is significantly improved. The CornerNet-Saccade with GCP outperforms that with CP by 6.5% when the number of arrayed objects in the same category is greater than 15.

We conduct ablation experiments on gradient corner pooling, handcrafted convolution kernel, and corner attention to evaluate their effects. As shown in Table 3, GCP encodes location information in homogeneous objects feature and addresses the information loss caused by directional max pooling. The handcrafted convolution kernels are used as filters to locate the corners in feature maps encoded by GCP. The combination of two parts is a complete pooling process and achieves the 0.7% and 0.9% improvements in AP and AR, respectively. Corner attention enables the backbone network to perform consistent feature learning with the prediction head, which improves the recall rate of the network. There is a boost of 2.1% in $AR_{100}$ with the corner attention mechanism and an improvement of 0.8% in $AP$.

| Numbers | w/ CP | | w/ GCP‡ | |
|---|---|---|---|---|
| | $AP$ | $AR_{100}$ | $AP$ | $AR_{100}$ |
| $> 5$ | 34.3 | 51.1 | 35.5 (+1.2) | 52.2 (+1.1) |
| $> 10$ | 32.7 | 47.3 | 34.7 (+2.0) | 50.6 (+1.2) |
| $> 15$ | 39.5 | 42.0 | 46.0 (+6.5) | 56.4 (+14.4) |

Table 2: Performance(%) comparison of CornerNet-Saccade on the arrayed objects subset of MS-COCO val2017.

| Method | Time | $AP$ | $AR_{100}$ |
|---|---|---|---|
| CornerNet-Saccade-54 | 145ms | 42.6 | 61.2 |
| replace CP with GCP | 145ms | 42.8 | 61.4 |
| + handcrafted Conv kernel | 146ms | 43.3 | 62.1 |
| replace attn with corner attn | 179ms | 44.1 | 64.2 |

Table 3: Ablation study(%) on gradient corner pooling, handcrafted convolution kernel, and corner attention in CornerNet-Saccade.

The patches of the image that need to be cropped which predicted by the attention mechanism are calculated in parallel on the GPU, the processing is efficient. In the inference process, the number of patches is dynamic and changes with the number of objects in the scene image. The threshold ($TH^{att}$) and the maximum number of attention locations are the hyperparameters of the attention mechanism. We measure the quality of the attention maps by average precision,

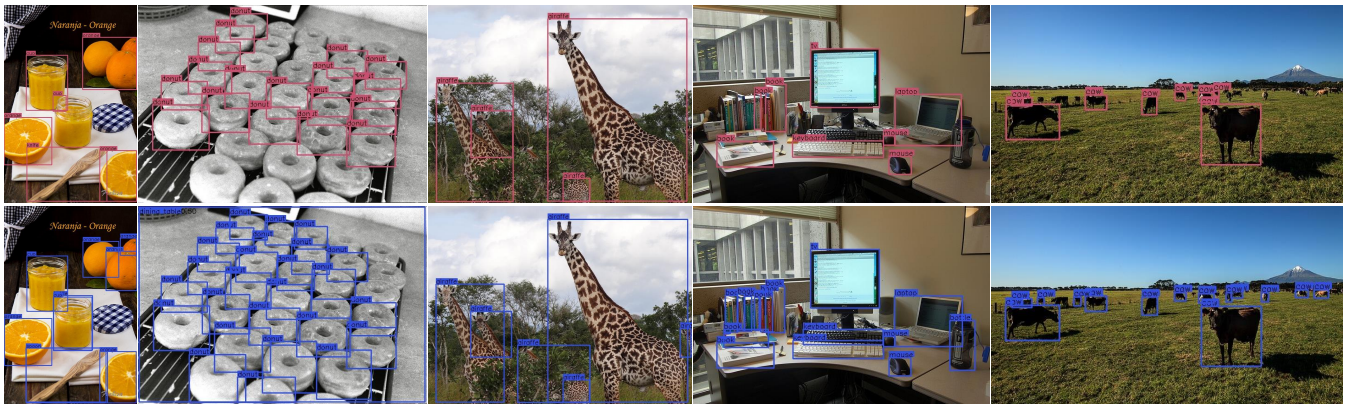


Figure 6: The first row in this figure is the detection result of CornerNet-Saccade with corner pooling while the second row is that with GCP module. The performance of keypoint-based detector is significantly improved with GCP for arrayed objects.

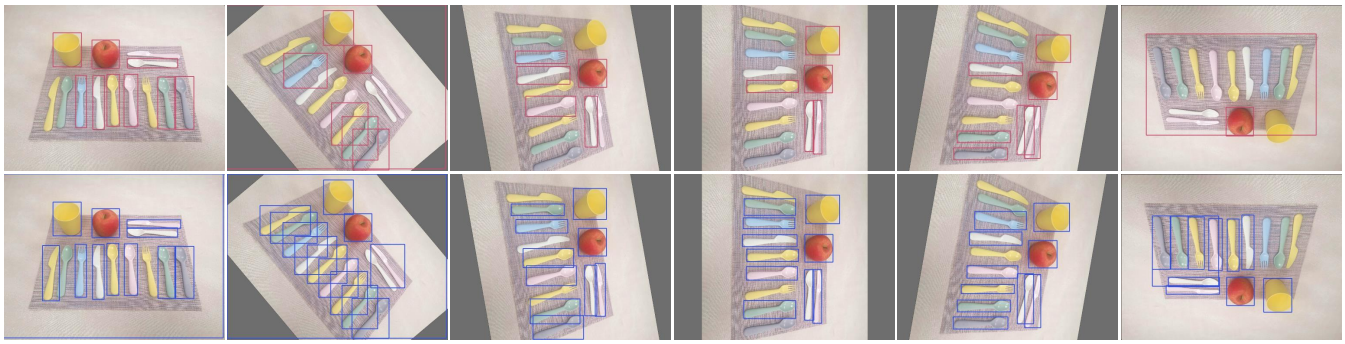



Figure 7: This is a picture taken in the real scene and then rotated clockwise by multiple angles. The detection results of the two rows of pictures are that the CornerNet-Saccade with the corner pooling and gradient corner pooling module, respectively. Since the top-left and bottom-right corners are far from the object itself for right oblique objects, corner detection becomes difficult. Identifying fixed geometric patterns by handcrafted convolution kernels fixes this problem well.

| Attention | $AP^{att}_{30}$ | $AP^{att}_{40}$ | $AP^{att}_{50}$ | $AP$ | $AR_{100}$ |
|---|---|---|---|---|---|
| Center | 52.0 | 52.0 | 52.0 | 42.6 | 61.2 |
| TL | 52.8 | 52.8 | 52.8 | 42.8 | 61.2 |
| BR | 53.1 | 53.0 | 53.0 | 42.8 | 61.0 |
| TL+BR | 54.6 | 54.8 | 54.8 | 44.1 | 64.2 |

Table 4: Comparison(%) of center attention and corner attention. The abbreviations are: 'TL' – top-left corners attention, 'BR' – bottom-right corners attention. The $AP$ and $AR_{100}$ are tested at $AP^{att}_{30}$ on MS-COCO val2017.

denoted as $AP^{att}$. We set the maximum number of patches that the attention mechanism is able to generate as 30, 40, and 50 respectively. There is little difference among $AP^{att}_{30}$, $AP^{att}_{40}$ and $AP^{att}_{50}$, indicating that the top 30 locations of $TH^{att} = 0.3$ are sufficient whether it is the center attention or the corner attention. As Table 4 shows, the $AP^{att}$ of top-left corner attention and bottom-right corner attention are both slightly higher than the center attention. The final $AP$ and $AR_{100}$ of corner attention have a better quality, suggesting that it is effective to enhance the consistency of backbone network and prediction head learning through GCP encoding attention.

## Conclusions

We propose gradient corner pooling (GCP) module and use it to build a corner attention mechanism. GCP encodes distance information in directional max pooling for the first time. These contributions refine the related theory of directional pooling and reveal the potential of a general directional max pooling for keypoint-based object detection.

## Acknowledgments

This work is supported by the National Key R & D Program of China (2020AAA0109301) and National Natural Science Foundation of China (61836008).

## References


Bochkovskiy, A.; Wang, C.-Y.; and Liao, H.-Y. M. 2020. YOLOv4: Optimal speed and accuracy of object detection. arXiv:2004.10934.

Cai, Z.; and Vasconcelos, N. 2018. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6154–6162.

Chen, Y.; Zhang, Z.; Cao, Y.; Wang, L.; Lin, S.; and Hu, H. 2020. Reppoints v2: Verification meets regression for object detection. *Advances in Neural Information Processing Systems*, 33: 5621–5631.

Dong, Z.; Li, G.; Liao, Y.; Wang, F.; Ren, P.; and Qian, C. 2020. Centripetalnet: Pursuing high-quality keypoint pairs for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10519–10528.

Duan, K.; Bai, S.; Xie, L.; Qi, H.; Huang, Q.; and Tian, Q. 2019. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 6569–6578.

Duan, K.; Bai, S.; Xie, L.; Qi, H.; Huang, Q.; and Tian, Q. 2022. CenterNet++ for object detection. arXiv:2204.08394.

Duan, K.; Xie, L.; Qi, H.; Bai, S.; Huang, Q.; and Tian, Q. 2020. Corner proposal network for anchor-free, two-stage object detection. In *European Conference on Computer Vision*, 399–416. Springer.

Ge, Z.; Liu, S.; Wang, F.; Li, Z.; and Sun, J. 2021. YOLOX: Exceeding YOLO series in 2021. arXiv:2107.08430.

Girshick, R. 2015. Fast r-cnn. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1440–1448.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask r-cnn. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2961–2969.

Law, H.; and Deng, J. 2018. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision*, 734–750.

Law, H.; Teng, Y.; Russakovsky, O.; and Deng, J. 2019. CornerNet-Lite: Efficient keypoint based object detection. arXiv:1904.08900.

Li, Y.; Chen, Y.; Wang, N.; and Zhang, Z. 2019. Scale-aware trident networks for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 6054–6063.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2980–2988.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *European onference on computer vision*, 740–755. Springer.

Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; and Berg, A. C. 2016. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, 21–37. Springer.

Lu, X.; Li, B.; Yue, Y.; Li, Q.; and Yan, J. 2019. Grid r-cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7363–7372.

Newell, A.; Yang, K.; and Deng, J. 2016. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, 483–499. Springer.

Papadopoulos, D. P.; Uijlings, J. R.; Keller, F.; and Ferrari, V. 2017. Extreme clicking for efficient object annotation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4930–4939.

Redmon, J.; Divvala, S.; Girshick, R.; and Farhadi, A. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 779–788.

Redmon, J.; and Farhadi, A. 2017. YOLO9000: better, faster, stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7263–7271.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

Tan, M.; Pang, R.; and Le, Q. V. 2020. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10781–10790.

Tian, Z.; Shen, C.; Chen, H.; and He, T. 2019. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9627–9636.

Wang, C.-Y.; Bochkovskiy, A.; and Liao, H.-Y. M. 2021. Scaled-yolov4: Scaling cross stage partial network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13029–13038.

Wang, C.-Y.; Bochkovskiy, A.; and Liao, H.-Y. M. 2022. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv:2207.02696.

Yang, Z.; Liu, S.; Hu, H.; Wang, L.; and Lin, S. 2019. Reppoints: Point set representation for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9657–9666.

Zhang, S.; Wen, L.; Bian, X.; Lei, Z.; and Li, S. Z. 2018. Single-shot refinement neural network for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4203–4212.

Zhou, X.; Wang, D.; and Krahenbuhl, P. 2019. Objects as points. arXiv:1904.07850.

Zhou, X.; Zhuo, J.; and Krahenbuhl, P. 2019. Bottom-up object detection by grouping extreme and center points. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 850–859.