Appendices

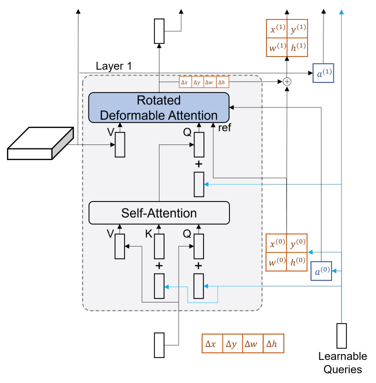
A. Implemenation Details of Our Baseline

In this section, we delve into the unique challenges associated with introducing rotation capabilities to DETR models and detail our specific implementation strategies.

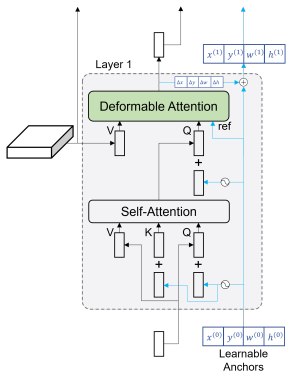
Angle Prediction by Simple Extended Box Regressor. In the DETRs, the coordinates of objects are handled as normalized ranges [0, 1] (relative to the image size) in the training stage. The normalized predicted coordinates from the sigmoid function are recovered to the original range by multiplying the width and height of the input image. One straightforward method to equip DETRs with the capability to predict an object's orientation is by extending the output of the regression head to $\mathbf{b} \in [0, 1]^5$, where $\mathbf{b} = [b_x, b_y, b_w, b_h, b_\theta]$ represents the normalized center coordinates, box width and height, and the angle in radians, respectively. Similar to the center coordinates and width and height, the angle should also be normalized. To facilitate this, we adopt the Long Edge Definition ($\theta \in [0, \pi)$) and normalize angle values to lie within the [0, 1] range by dividing the ground truth angles by π. During the training phase, the rotated DETR is trained to predict these normalized coordinates. At the inference time, the predictions are converted back to their original scale by multiplying the output $\mathbf{b}$ with $[I_w, I_h, I_w, I_h, \pi]$, where I_w and I_h denote the width and height of the image, respectively. By treating angles the same way as other coordinate values, rather than introducing a dedicated angle prediction branch, the model can easily incorporate other developed components related to coordinates, such as iterative bounding box refinement, handling queries as anchor boxes, and proposing queries from the feature map of the encoder.

Replacement of GIoU Loss. The standard costs and losses used in DETRs are Focal loss, L1 loss, and GIoU loss. However, calculating overlaps between two rotated objects is commonly known to be indifferentiable [11, 12, 34]. Therefore, we need to consider alternative options to replace GIoU loss for rotated objects. Two representative substitutes for representing the error between predictions and ground truth for rotated objects are Gaussian Wasserstein Distance (GWD) [12] and Kullback-Leibler Divergence (KLD) [11]. The center distance term in GWD and KLD also works as the penalty term for non-overlapping bounding boxes, similar to GIoU loss. We adopt KLD as our baseline to replace the GIoU cost and loss since previous work has shown that KLD outperforms GWD in oriented object detection. Additionally, we set the weight for the IoU term from 2 to 5 based on the comparison experiment, which is shown in Table 8a. This weight is applied to all our experiments except the experiments conducted on the patch validation set.

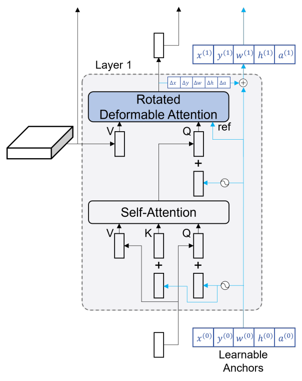
Rotated Deformable Attention. In order to address the slow convergence and high computational memory complexity associated with the multi-head attention module in the


Layer 1
Rotated Deformable Attention
V
Q
ref
Self-Attention
V
K
Q
$x^{(1)}$ $y^{(1)}$ $w^{(1)}$ $h^{(1)}$
$a^{(1)}$
$x^{(0)}$ $y^{(0)}$ $w^{(0)}$ $h^{(0)}$
$a^{(0)}$
Δx Δy Δw Δh
Learnable Queries

(a) ARS-DETR


Layer 1
Deformable Attention
V
Q
ref
Self-Attention
V
K
Q
$x^{(1)}$ $y^{(1)}$ $w^{(1)}$ $h^{(1)}$
$x^{(0)}$ $y^{(0)}$ $w^{(0)}$ $h^{(0)}$
Learnable Anchors

(b) DAB-DETR


Layer 1
Rotated Deformable Attention
V
Q
ref
Self-Attention
V
K
Q
$x^{(1)}$ $y^{(1)}$ $w^{(1)}$ $h^{(1)}$ $a^{(1)}$
$x^{(0)}$ $y^{(0)}$ $w^{(0)}$ $h^{(0)}$ $a^{(0)}$
Learnable Anchors

(c) Ours

Figure 5. Comparison of attention layers in different models. (a) ARS-DETR, based on Deformable DETR, predicts reference boxes from learnable queries but does not fully integrate angle information into the objects' spatial information. (b) DAB-DETR, which our model builds upon, defines learnable queries as horizontal anchors. (c) Our model extends this approach by defining anchors as rotated boxes, leveraging methods such as iterative refinement and two-stage proposals from DINO, resulting in improved integration of angle information.

original DETR, Zhu et al. [19] proposed a deformable attention module. This module takes an input feature map and either 2D reference points $\mathbf{p_2} = (p_x, p_y)$ or 4D reference boxes $\mathbf{p_4} = (p_x, p_y, p_w, p_h)$, where (p_x, p_y) represents a series of center coordinates and (p_w, p_h) represents a series of widths and heights for the reference boxes, to determine the sampling location. The sampling locations for 2D reference points are obtained by adding the predicted sampling offsets to the given reference points. In the case of 4D reference boxes, the predicted sampling offsets are first added to the center coordinates of reference boxes. Then, the resulting values are multiplied by half of the widths and heights of the reference boxes to determine the final 2D coordinates for the sampling locations. Using these 4D reference boxes enhances performance through iterative refinement and query proposal from the encoder.

As Zhu et al. [28] pointed out, using $\{b_x, b_y, b_w, b_h\}$ as reference boxes from 5D reference boxes by dropping b_{rad} leads to feature misalignment, especially in deformable attention modules. We apply a similar way to ARS-DETR [28] to rotate the center coordinates from the reference boxes. The predicted normalized angles are recovered by multiplying the normalized factor π, and the next steps follow the case of 4D reference boxes as described previously. In contrast to the approach proposed by Zeng et al. [28], we update 5D reference boxes including angles during the iterative refinement step. To update the 5D reference boxes, we apply the $\mathrm{inverse\ sigmoid}$ function to the reference points and add the resulting values to the logit of the regression head in logit space. We then normalize the updated values using the $\mathrm{sigmoid}$ function, which maps them to the range of [0, 1]. The same update logic is also applied to the predicted regression in the head part. A visualization of our deformable attention is illustrated in Figure 5. Building on the principles of DAB-DETR [20], which defines learnable queries as dynamic anchor boxes, our model integrates angles into oriented reference boxes in a natural way.

Extension to DINO. By straightforwardly extending the regression head with 5D rotated boxes, the model is able to naturally adopt the developed mechanisms such as queries as dynamic anchor boxes proposed by successive works [20, 21, 39]. Here, we just brief some implementation details to note. The two-stage scheme introduced by Deformable-DETR [19], requires **generating grid proposals from the encoded features**, and predicting query proposals by running the regression head to the encoded features. In our implementation, We use zero-angle grids to generate grid proposals from the encoded features and run the regression head on the flattened feature vectors. This allows the regression output to be added directly to the proposals. For **denoising training** [21, 39], we add noises only to $\{b_x, b_y, b_w, b_h\}$ for simplicity. We exclude the angle for the decoder positional queries which are related to **mixed query selection** in DINO [21]. Finally, we experiment with models using 900 queries and set the maximum number of predictions to 500 to handle densely packed objects in remote sensing images.

B. Implementation Details of RHINO

We implement our models on MMRotate [52] v1.0.0rc1. Most of our experiments were conducted on 2 NVIDIA V100 or A100 GPUs, with a total batch size of 8. Two backbones: ResNet-50 [53] and Swin-Tiny [54] are used in our experiments, both of which are pre-trained on ImageNet-1k [55]. We use the AdamW [56, 57] optimizer with an initial learning rate of 1×10^{-4}. We use a step decay learning rate schedule where the learning rate is multiplied by 0.1 at the 11th epoch during the 12 epoch training and multiplied by 0.1 again at the 27th and 33rd epochs during the 36 epoch training. For DOTA-v1.0/v1.5/v2.0, we crop each image in the data sets to 1024×1024 pixels with a 200-pixel overlap. The input size of experiments on DIOR-R is set to 800×800. We use only horizontal, vertical, and diagonal flips without additional augmentations.

C. More Ablation Studies on Model Components.

Effect of the number of points for the Hausdorff distance. In our approach, we approximate the Hausdorff distance between two rotated boxes using points along the box edges. As illustrated in Figure 6, increasing the number of points improves the precision of matching, as reflected in the IoU. To assess the effect of point count, we evaluated different configurations, with the results summarized in Table 8b. The 4-point configuration achieved the highest performance in AP50, with a score of 78.68. However, the 32-point configuration yielded the best result in AP75, achieving 51.84. Since AP75 uses a stricter IoU threshold of 0.75 compared to 0.5 in AP50, this indicates that increasing the number of points improves performance in more precise matching evaluations. Nevertheless, for simplicity and efficiency, we use 4 points to compute the Hausdorff distance in all other experiments.

Updating angles in logit space. Table 8c presents the importance of updating angles in DETR components. **IR** and **Two-Stage** represent iterative bounding box refinement and a two-stage approach, respectively, as introduced by Deformable DETR, but without incorporating the angle parameter. **5D RP** refers to the updating angles of 5D reference points in Decoder layers, while **5D Head** refers to the updating angles of predictions using reference points in the head module. No checkmark in **5D RP** or **5D Head** columns indicates that only the coordinates $\{b_x, b_y, b_w, b_h\}$ of the reference points or predictions are updated. This finding illustrates that treating oriented object detection as a regression problem benefits from adopting the latest query-as-bounding-

Table 8. Further ablation studies on DOTA-v1.0.

(a) Robustness on weight for IoU term

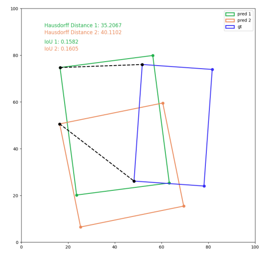
$\mathcal{L}_{L1}$ Cost	$\mathcal{L}_{L1}$ Loss	λ_{iou}	AP_{50}	w NMS
None	None	2	68.30	+0.10
None	None	5	70.18	-0.05
Hausdorff	L1	2	70.81	-0.09
Hausdorff	L1	5	70.94	+0.16

(b) The number of points for the Hausdorff distance

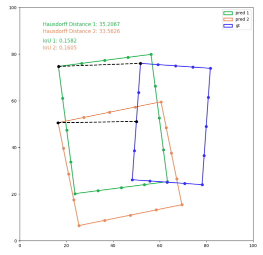
# points	AQD*	AP_{50}	AP_{75}
4	✓	78.68	51.17
8	✓	78.12	51.02
32	✓	78.49	51.84

(c) Effect of updating angles in the baseline

Model	Epochs	IR	Two-Stage	5D RP	5D Head	AP_{50}
Deformable DETR	50	-	-	-	-	68.50
Deformable DETR	50	✓	-	-	-	68.54
Deformable DETR	50	✓	✓	-	-	70.48
DINO	12	✓	✓	-	-	71.36
DINO	12	✓	✓	✓	-	72.76
DINO	12	✓	✓	✓	✓	76.10



(a) Hausdorff distance with 4 points.



(b) Hausdorff distance with 16 points.

Figure 6. Visualization of the Hausdorff distance for different numbers of points. Matching is based on the smaller Hausdorff distance, with higher IoU indicating better matching.

box approaches proposed in state-of-the-art DETR models.

Analysis of the impact of aspect ratios. To evaluate the effectiveness of the Hausdorff distance in addressing the issue of duplicate predictions, we conducted a class-wise performance comparison using the DOTA-v1.0 patch validation set. The results, detailed in Table 10, reveal that using the Hausdorff distance significantly improves performance for square-like objects, such as `baseball-diamond`, `storage-tank`, and `roundabout`.

D. Computational Cost Comparison

Table 9 presents a performance comparison of different methods. Note that DINO requires more computation due to its Transformer modules. Using the Hausdorff distance (with 4 corner points) enhances performance without significantly increasing computation. The adaptive query denoising method incurs higher computational costs because it involves additional bipartite matching during the denoising process. However, the inference speed and number of parameters remain unchanged compared to DINO, as the proposed methods primarily affect model training.

E. Experiments on Other Rotated Object Datasets

We extend the evaluation of our model, RHINO, by comparing it with the DINO baseline across various other rotated object detection datasets. Specifically, we focus on two scene text detection datasets, MSRA-TD500 and ICDAR2015, as well as the retail object dataset SKU110K-R. For the MSRA-TD500 dataset, we trained both RHINO and DINO models for 50 epochs. For the SKU110K-R and ICDAR2015 datasets, the training duration was set to 36 epochs. As detailed in Table 11, RHINO consistently outperforms the DINO baseline, demonstrating its robustness and effectiveness even though these datasets generally include non-square objects.

Table 12 presents a comparison between our model and state-of-the-art models on the HRSC2016, which is a widely used aerial ship dataset for oriented object detection. While our model, RHINO, exhibits marginally lower performance than the state-of-the-art models in terms of mAP_{07}, it is noteworthy that RHINO surpasses all other models in mAP_{12}.

F. Visualization

Figure 7 presents a qualitative comparison of our models and other models. Notably, the RoI Transformer demonstrates lower precision in predicting object angles compared to our model and ARS-DETR.

Figure 8 and Figure 9 provide further comparisons between the baseline and our model. Overall, RHINO exhibits relatively consistent non-duplicate predictions compared to the baseline, especially in square-like objects, as shown in Figure 9. However, it is important to note that RHINO occasionally produces duplicate predictions on the MSRA-TD500 dataset. We assume this is due to the small amount of training dataset for MSRA-TD500.

Table 9. Computational cost comparison using 2 A-40 GPUs on DOTA-v1.0

Method	Params	GFLOPs	$FPS_{bs=1}$	Inf. Memory	Train Time	Train Memory	AP_{50}
DCFL [7]	36.1M	157.80	23	2.3 GB	11 Hours	4 GB	75.35
ARS-DETR [28]	41.6M	205.95	14	2.2 GB	24 Hours	10-11 GB	73.42
DINO	47.3M	280.38	13	2.2 GB	25 Hours	16-20 GB	74.56
+ Hausdorff dist.	47.3M	280.38	13	2.2 GB	25 Hours	16-20 GB	76.14
+ AQD*	47.3M	280.38	13	2.2 GB	30 Hours	16-20 GB	78.68

Table 10. Class-wise performance and its average aspect ratios on the patch validation set.

class	Aspect Ratio	L1	Hausdorff
plane	0.7982	89.0	89.8
baseball-diamond	**0.9239**	**67.1**	**71.5**
bridge	0.4350	48.6	50.9
ground-track-field	0.5044	66.3	66.8
small-vehicle	0.4736	69.2	69.1
large-vehicle	0.2746	84.5	83.7
ship	0.3535	87.7	88.4
tennis-court	0.4828	90.6	90.5
basketball-court	0.5748	60.0	56.7
storage-tank	**0.9365**	**63.3**	**73.1**
soccer-ball-field	0.6109	56.2	58.2
roundabout	**0.9224**	**52.3**	**60.8**
harbor	0.4018	75.8	76.5
swimming-pool	0.6087	53.7	54.9
helicopter	0.7146	55.5	71.4
AP_{50}	0.4705	68.0	70.81

Table 11. Performance comparison on rotated object datasets

Method	Dataset	AP_{50}	$Hmean_{50}$
DINO	SKU110K-R	87.22	-
RHINO	SKU110K-R	88.30	-
DINO	MSRA-TD500	-	0.5862
RHINO	MSRA-TD500	-	0.6173
DINO	ICDAR2015	-	0.5685
RHINO	ICDAR2015	-	0.5819

Table 12. Performance comparison on HRSC2016.

Method	Backbone	mAP_{07}	mAP_{12}
RoI Trans. [3]	R-101	86.20	-
R^3Det [58]	R-101	89.26	96.01
S^2ANet [8]	R-101	90.17	95.01
Oriented R-CNN [10]	R-50	90.40	96.50
ReDet [9]	ReR-50	90.46	97.63
RTMDet-R-tiny [46]	CSPNeXt-tiny	**90.60**	97.10
DINO [21] (Our implementation)	R-50	90.26	97.37
RHINO (Ours)	R-50	90.30	**97.86**

G. Limitation

Despite the significant performance improvements achieved by our model, some limitations exist. The adaptive query denoising method, which leverages bipartite matching to filter out harmful noised queries selectively, necessitates extended training times, as shown in Table 9. Furthermore, while adaptive query denoising consistently enhances performance across various rotated object detection tasks, its application to the DINO model trained on the COCO dataset results in a slight performance decrement (-1.6 AP) compared to the baseline. We suspect this reduction in performance may stem from the observation that the denoising task for horizontally aligned objects is less impacted by noisy queries, unlike their rotated counterparts. Future work may explore more improved methods to adaptively control denoising tasks, potentially bypassing the need for bipartite matching. such as the adaptive matching cost for denoising. This might include developing an adaptive matching cost specifically tailored for the denoising task, which could offer a more nuanced approach to improving model performance while mitigating extended training times.

Algorithm 1 Contrastive Query Denoising of DINO [21].

```
Input: Predicted objects $\mathbf{p} = \{p_0, p_1, \dots, p_{N-1}\}$, Ground truths $\mathbf{y} = \{y_0, y_1, \dots, y_{M-1}\}$, Positive noised queries $\mathbf{q}^{\text{pos}} = \{q_0^{\text{pos}}, q_1^{\text{pos}}, \dots, q_{M-1}^{\text{pos}}\}$, Negative noised queries $\mathbf{q}^{\text{neg}} = \{q_0^{\text{neg}}, q_1^{\text{neg}}, \dots, q_{M-1}^{\text{neg}}\}$.

1: Initialization: $\mathbf{p}^{\text{pos}} = f_{\text{refine}}(\mathbf{q}^{\text{pos}})$, $\mathbf{p}^{\text{neg}} = f_{\text{refine}}(\mathbf{q}^{\text{neg}})$.
   Calculate the loss: $\mathcal{L}_{\text{denoising}}(\mathbf{p}^{\text{pos}}, \mathbf{p}^{\text{neg}}, \mathbf{y}) = \sum_{i=0}^{M-1} \mathcal{L}_{\text{pos}}(p_i^{\text{pos}}, y_i) + \mathcal{L}_{\text{neg}}(p_i^{\text{neg}}, \varnothing)$.
   Match Queries and Predictions
2: Set assignment $\sigma$ with the corresponding ground truths.
3: $\sigma = 0, 1, \dots, M-1$.
   Filter Unhelpful Queries:
4: $\mathcal{L}_{\text{pos}}(p_i^{\text{pos}}, y_i) = \mathbb{1}_{\{\sigma_i = i\}} \mathcal{L}_{\text{train}}(p_i^{\text{pos}}, y_i)$.
5: Update: Perform backpropagation and update model parameters.
```

Algorithm 2 Our Improved Adaptive Query Denoising. The blue text lines represent the modifications and highlight the differences from Algorithm 1.

```
Input: Predicted objects $\mathbf{p} = \{p_0, p_1, \dots, p_{N-1}\}$, Ground truths $\mathbf{y} = \{y_0, y_1, \dots, y_{M-1}\}$, Positive noised queries $\mathbf{q}^{\text{pos}} = \{q_0^{\text{pos}}, q_1^{\text{pos}}, \dots, q_{M-1}^{\text{pos}}\}$, Negative noised queries $\mathbf{q}^{\text{neg}} = \{q_0^{\text{neg}}, q_1^{\text{neg}}, \dots, q_{M-1}^{\text{neg}}\}$.

1: Initialization: $\mathbf{p}^{\text{pos}} = f_{\text{refine}}(\mathbf{q}^{\text{pos}})$, $\mathbf{p}^{\text{neg}} = f_{\text{refine}}(\mathbf{q}^{\text{neg}})$.
   Denoising Loss: $\mathcal{L}_{\text{denoising}}(\mathbf{p}^{\text{pos}}, \mathbf{p}^{\text{neg}}, \mathbf{y}) = \sum_{i=0}^{M-1} \mathcal{L}_{\text{pos}}(p_i^{\text{pos}}, y_i) + \mathcal{L}_{\text{neg}}(p_i^{\text{neg}}, \varnothing)$.
   Match Queries and Predictions
2: $\mathbf{p}^{\text{all}} = [\mathbf{p}^{\text{pos}}; \mathbf{p}] = [p_0^{\text{pos}}, p_1^{\text{pos}}, \dots, p_{M-1}^{\text{pos}}, p_0, p_1, \dots, p_{N-1}]$.
3: Find optimal assignment $\sigma$ using bipartite matching.
4: $\hat{\sigma} = \text{argmin}_{\sigma \in \mathfrak{S}} \sum_{i=0}^{M+N-1} \mathcal{L}_{\text{match}}(\hat{y}_i, \mathbf{p}_{\sigma_i}^{\text{all}})$.
   Filter Unhelpful Queries:
5: $\mathcal{L}_{\text{pos}}(p_i^{\text{pos}}, y_i) = \mathbb{1}_{\{\hat{\sigma}_i = i\}} \mathcal{L}_{\text{train}}(p_i^{\text{pos}}, y_i) + \mathbb{1}_{\{\hat{\sigma}_i \neq i\}} \mathcal{L}_{\text{neg}}(p_i^{\text{pos}}, \varnothing)$.
6: $\mathcal{L}_{\text{pos}}^{*}(p_i^{\text{pos}}, y_i) = \mathcal{L}_{\text{pos}}(p_i^{\text{pos}}, y_i) + \mathbb{1}_{\{\hat{\sigma}_i \neq i\}} \mathcal{L}_{\text{bbox}}(p_i^{\text{pos}}, y_i)$.
7: Update: Perform backpropagation and update model parameters.
```

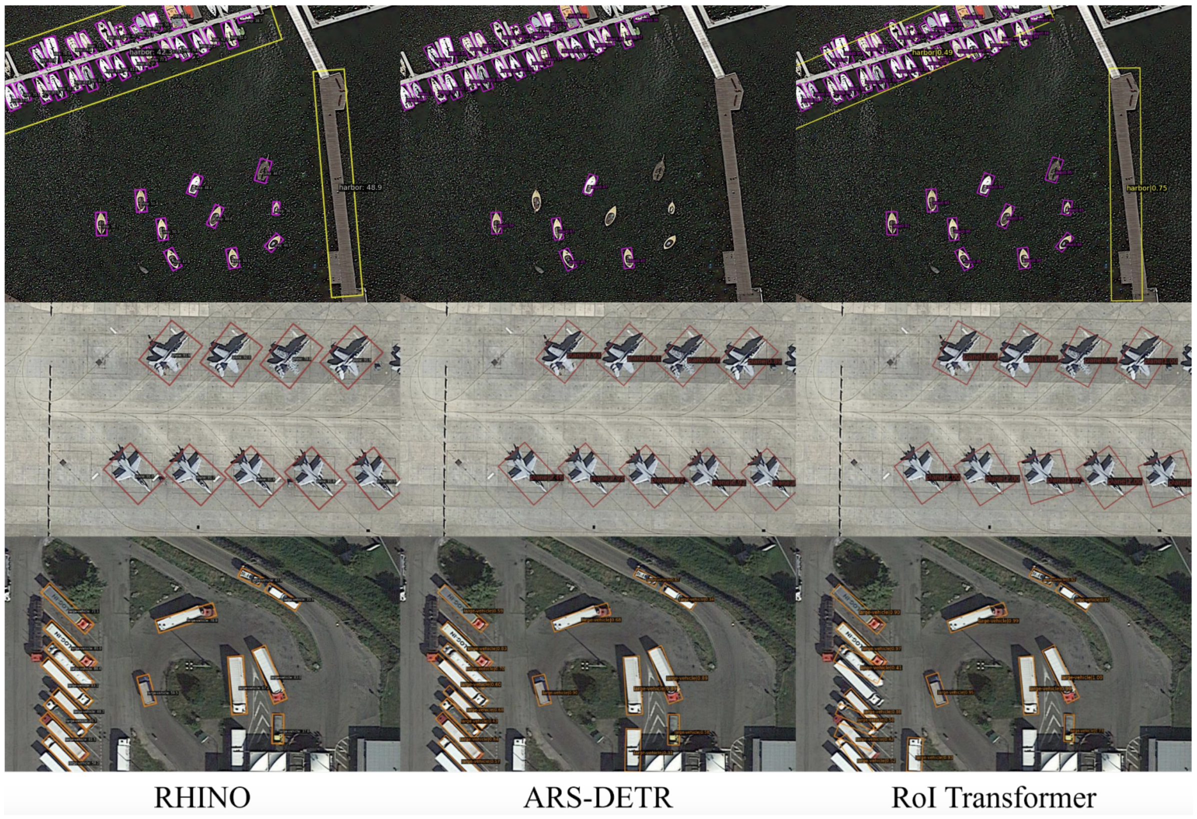


Figure 7. Qualitative comparison between our model and other models on the DOTA-v1.0 dataset.

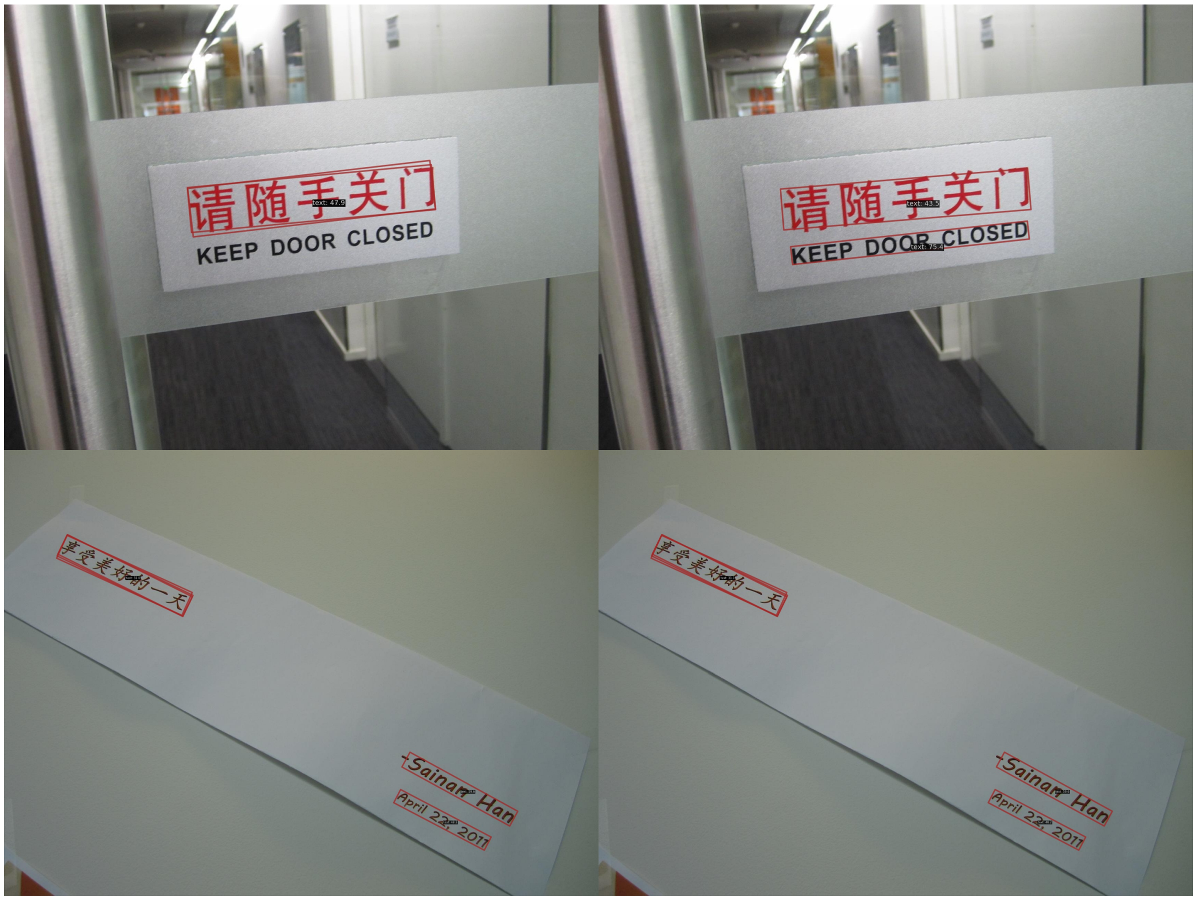


Figure 8. Qualitative comparison between the baseline and our model on the MSRA-TD500 dataset.

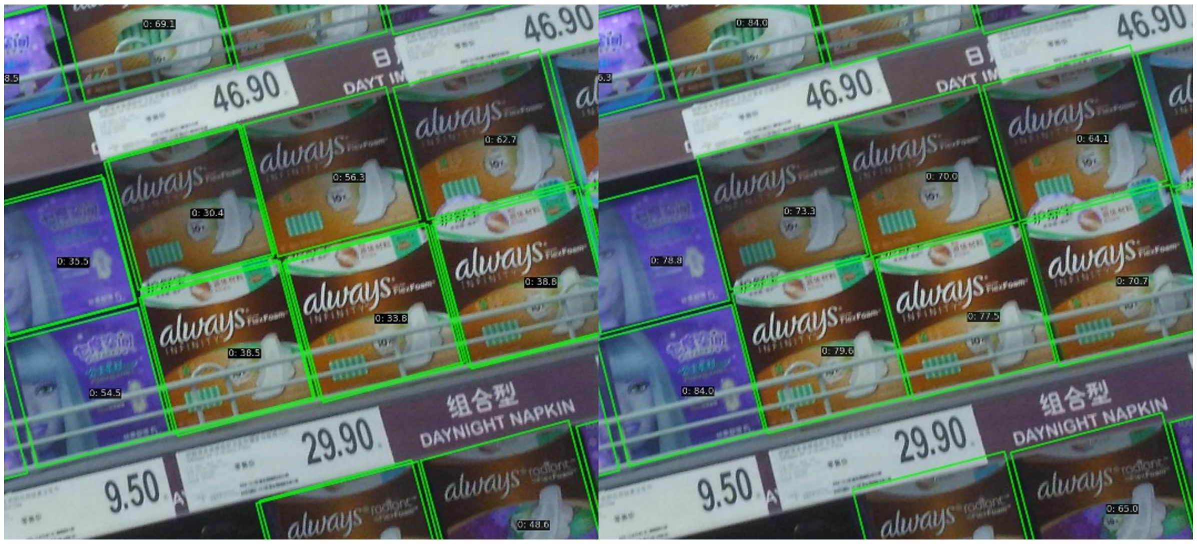


Figure 9. Qualitative comparison between the baseline and our model on the SKU110K-R dataset.